Pablo Rodríguez

# Aprender ConTEXt

*Una introducción a la tipografía digital*



http://www.aprender-context.tk

# *Aprender ConTeXt*

## *Una introducción a la tipografía digital*

Pablo Rodríguez

2018

http://www.aprender-context.tk

*Para Edu,*
*ya no hay excusas.*

# SUMARIO

## INTRODUCCIÓN



## ANTES DE EMPEZAR



## PRIMEROS PASOS



## APÉNDICES

# PRÓLOGO

Este libro pretende ser una introducción en español a ConTEXt. Su punto de partida son los conocimientos que tiene cualquiera que se acerque por primera vez a este tipo de programas: ninguno o casi ninguno. El libro procura explicar todo de modo gradual, sin dar nada por supuesto. No supone nada más que quien lo lee sabe usar un ordenador en sus funciones básicas. Basta con saber usar un procesador de textos.

En la medida en que es una explicación de ConTEXt, este libro es una introducción a la tipografía digital. No es en absoluto un tratado sobre tipografía digital, para el que seguro me faltarían conocimientos. No pretendo tal cosa. Se trata de *una* introducción a la tipografía digital en la medida en que se explica a propósito de la descripción de las capacidades de ConTEXt.

El libro está dividido en tres grandes apartados: una introducción relativamente extensa, el contenido del libro propiamente dicho y los apéndices. La clave del libro está en la parte introductoria, que es relativamente larga. La introducción es importante en un doble sentido, por ConTEXt y por el libro. Para que de la mejor manera, quien lea sepa a qué atenerse. Recomiendo leer la introducción con calma y pensarla antes de empezar con el resto del libro. Para saber qué es lo que hay, qué es lo que buscamos, y qué es lo que nos podemos encontrar.

25 de junio de 2017

# INTRODUCCIÓN

# CONTEXT

*¿Qué es ConTEXt? Orígenes ¿Por qué ConTEXt?*

## ¿QUÉ ES CONTEXT?

ConTEXt es un sistema de generación automatizada de documentos. Esto quiere decir que permite componer documentos con alta calidad tipográfica. Estos documentos pueden ser publicaciones impresas o digitales. De un archivo de origen con la información adecuada, genera documentos PDF. También puede crear documentos XML. El archivo de origen tiene que contener el texto con las órdenes de composición tipográfica[1]. En resumen, ConTEXt es un programa con el que podemos generar directamente archivos PDF de alta calidad tipográfica.

La clave de ConTEXt no está en que sea un programa que genere documentos PDF. Sobre todo porque existen ya unos cuantos programas que realizan esa función. La característica esencial de ConTEXt es que, al ser un programa de composición tipográfica digital[2], puede generar documentos que pueden publicarse —tanto en papel, como electrónicamente— con una calidad profesional. Como si estuviesen compuestos para la imprenta tradicional.

Con una imagen que analizaremos más tarde, ConTEXt es una imprenta en tus manos[3]. Un procesador de textos —como puede ser *LibreOffice Writer*— es una máquina de escribir. Sin duda, una máquina de escribir es más fácil de usar que una imprenta. Sin embargo, los resultados de una máquina de escribir, aunque siendo correctos, son insuficientes cuando se comparan con la imprenta. De otro modo, la mecanografía habría sustituido a la imprenta.

ORÍGENES

Como programa —ésta es una explicación un poco más técnica[4]—, ConTEXt es en realidad un conjunto de instrucciones que usan un lenguaje de programación tipográfica. No es un programa que usa otro programa. Es una simplificación en el uso unas capacidades muy precisas para la persona que ha de realizar la tarea de tipógrafo digital.

El lenguaje de programación tipográfica es también un sistema de composición tipográfica digital, llamado TEX. Está desarrollado por Donald Knuth, catedrático emérito de la Universidad de Stanford. Originariamente, Knuth quería conseguir mayor calidad en la publicación de sus libros, que había bajado considerablemente con la introducción de la informática. Muchas fórmulas matemáticas estaban especialmente mal compuestas en las galeradas hechas con ordenadores.

Knuth es una autoridad mundial en teoría de programación informática. En 1976, al volver a editar el segundo volumen de su obra *The Art of Computer Programming*[5], había que volverlo a componer tipográficamente. Las linotipias ya no eran una opción y había que usar fotocomposición con ordenadores. Cuando los editores le presentaron las primeras pruebas de imprenta, el resultado le pareció horroroso. De hecho, había publicado ya tres volúmenes y tenía más que publicar. Aunque, si el resultado era ése, no tenía ganas de seguir escribiéndolos[6]. En 1977, decidió tomarse un tiempo para desarrollar un sistema de composición tipográfica digital.

En 1989, Knuth anunció que el sistema de composición tipográfica TEX era estable. Sólo corregiría fallos graves. TEX tiene una licencia que permite su modificación. La única condición es que el programa derivado no use el mismo nombre. El mismo Knuth ha animado a otras personas a que amplíen las posibilidades tipográficas para nuevos proyectos. Entre otros muchos, cabe destacar LuaTEX, el derivado de TEX que ConTEXt usa.

ConTEXt nació de la necesidad de tener un conjunto de órdenes que permitiese usar TEX de un modo más fácil y que hiciese posible además tareas que TEX no realiza. Está desarrollado desde 1990 por Hans Hagen. Propiamente ConTEXt es un conjunto de macroinstrucciones que permiten a LuaTEX componer tipográficamente un texto. Escrito en español corriente, son un conjunto de órdenes que permiten a un programa especializado generar documentos PDF de alta calidad de modo totalmente automatizado. Además, también permite generar propiedades interactivas en los archivos PDF. Entre otros, hiperenlaces[7], anotaciones digitales o formularios para rellenar en el mismo documento.

La descripción del párrafo anterior no da razón de todas las posibilidades de ConTEXt. No pretendo hacer un análisis completo, porque sería fácil que agobiase a cualquiera. La idea básica es que se trata de un programa de composición tipográfica automatizada. Le dices qué es cada parte del texto, cómo tiene que ir en la página final, y ConTEXt coloca cada cosa en su sitio. Nunca he usado *InDesign*, pero intuyo que con el programa de la casa *Adobe*, cada cosa la tienes que colocar tú en el sitio que quieras que vaya. Eso será importante para el diseño gráfico de pubicaciones ilustradas, donde todo está supeditado a la primacía de la imagen. Pero no es ninguna ventaja para la composición digital de textos.

Como comentario absolutamente personal, las características que más destacaría en este momento son dos. La primera virtud es su licencia y desarrollo abiertos. Eso se traduce en varios aspectos. El primero y no menor es que si hubiese tenido que pagar una única licencia de *InDesign*, me temo que nunca lo hubiese usado. Sencillamente, no podría pagármelo. ConTEXt lo tengo instalado —lo empleo a diario— en todos los ordenadores que uso. Tanto en casa, como en el trabajo. El desarrollo abierto también se traduce en la posibilidad de que se corrijan errores en horas o días. Lo mismo sucede con las sugerencias de mejora

motivadas y relevantes. La segunda joya es que sea multiplataforma. Tengo *Linux* en casa y *Windows* en el trabajo. No comparto archivos o código, pero tengo una sola cabeza. Tengo amigos que usan *MacOS X*. Es una solución de la que puedo servirme y con la que puedo ayudar a todos. Cada día que pasa valoro más esa característica que podría parecer secundaria.

### ¿POR QUÉ CONTEXT?

Es posible que a muchas personas, tras hojear este libro —o incluso después de leerlo—, puedan llegar a la conclusión de que ConTEXt es sólo un modo de complicar innecesariamente las cosas. Tengo que reconocer que estoy absolutamente de acuerdo con esa concepción. Quien piense sinceramente que la tipografía digital es sólo una manera de complicar las cosas, debe hacerlas de modo que considere más sencillo. ConTEXt no puede ser una herramienta útil para esa persona, porque realmente su tarea es otra.

Por si el párrafo anterior parece contradictorio con la finalidad del libro, creo que es importante aclarar una cuestión básica. No existe algo así como la utilidad universal. Útil es siempre útil para algo. La utilidad es siempre relacionada con el fin que se persigue. De ahí que lo que puede ser muy útil para lograr un objetivo, puede ser incluso un impedimento para lograr otro. Antes de examinar si un medio o un instrumento —en este caso, digital— es útil, tenemos que saber cuál es la tarea que tenemos entre manos. No es una obviedad, hemos de entender qué tenemos que alcanzar.

Al punto anterior, es necesario añadirle otra precisión. Supongo que es será algo obvio para toda persona que pueda llegar a estas páginas, pero de todas formas lo abordo. Este libro no trata de motivar a nadie para que haga nada. Al menos entre personas adultas, tengo el convencimiento de que la motivación no la puede generar más que

cada cual para sí. Espero poder llegar a mostrar lo interesante de las posibilidades de ConTEXt. Aunque incluso el interés que tenemos por las cosas es algo personal e intransferible.

Mi intención no es tratar de convencer de que todo el mundo tiene que usar ConTEXt. Estoy casi convencido de que la afirmación anterior es muy probablemente falsa. Desde luego, no seré yo quien tenga los argumentos para defenderla. Considero que hay argumentos muy interesantes en favor de la utilidad de ConTEXt. Vuelvo a afirmar que no existe la utilidad universal. Porque útil es siempre para algo.

ConTEXt no es un procesador de textos. Ni puede ni pretende serlo, porque sencillamente es algo distinto. No es mejor ni peor que *Microsoft Word* o *LibreOffice Writer* —por poner dos ejemplos—, realmente se trata de otra cosa. Aunque todos sean programas informáticos que trabajan con textos. Un procesador de texto no deja de ser una máquina de escribir con memoria. Ésa es una grandísima utilidad para un número grandísimo de personas. Sencillamente se teclea y en pantalla aparece lo tecleado con la visión final que va a tener el texto, de igual modo que lo haría la hoja mecanografiada en una máquina de escribir.

Un sistema de composición tipográfica de documentos funciona de otra manera. Los elementos son el texto hay que componer y las órdenes de composición automática del texto —el formato—. Cuando la persona lo decide, ordena al programa que aplique las órdenes de composición y compile el documento compuesto tipográficamente. Así se genera el documento final. Sería como llevar a imprenta el manuscrito para que un tipógrafo lo componga.

El procesador de textos se basa un enfoque visual, que viene a ser la hoja en blanco en la que se escribe. Por supuesto, esto lo simula el ordenador para que la persona que lo use no vea diferencia irreconciliable respecto a trabajar sin un ordenador.

El sistema de composición tipográfica usa un enfoque lógico. No simula que tuviésemos una hoja en blanco en el carro de una máquina de escribir. Exige que le definan el texto y sus diferentes elementos y que le especifiquen las condiciones con las que se quiere componer el texto. Como si se tratase de un libro que está en la imprenta.

Para quienes no hayan manejado más que procesadores de texto, renunciar al enfoque visual les puede costar un esfuerzo considerable. Sin embargo, el enfoque lógico tiene varias ventajas.

En ConTEXt todo es inmediatamente visible y todo está a la vista. Esto es muy importante para saber qué pasa y por qué nos falla lo que nos está fallando. Cuando el documento no es completamente visible, no es fácil saber muchas veces si el fallo está en el programa o en el documento —en un error de la persona—.

Puesto que ConTEXt guarda los datos de los documentos que tiene que componer en archivos de texto puro, es mucho más fácil poder abrir y modificar esos archivos. Pero no sólo eso. Es mucho más fácil saber con toda precisión saber qué es exactamente lo que ha cambiado de un archivo a otro. Pero sobre todo, las modificaciones entre dos versiones de un mismo archivo. Porque en un procesador de textos hay muchas cosas que el programa sencillamente no muestra.

Debido a que los archivos que usa ConTEXt son transparentes —en el sentido descrito en los dos párrafos anteriores— es muy fácil llevar un control de cambios con un programa específico[8]. El control, al tratarse de archivos de texto puro, es total.

Y en el fondo, el enfoque lógico es el que permite acometer de modo más sencillo, más rápido y más eficiente tareas que son muchísimo más complicadas usando un enfoque visual. Pongo un ejemplo con el que he hecho pruebas: la composición de ediciones críticas[9]. No hace falta saber filología para advertir que las ediciones críticas son textos con estructura visual compleja. Una edición crítica no especialmente

compleja puede tener dos aparatos. Esto supone un texto con dos cuerpos de notas, en los que no hay números de notas al pie, sino que sólo hay referencias en los cuerpos de notas a las líneas del texto. Ése es el resultado final.

Si en el proceso de generación de un documento complejo, la parte lógica no es lo más sencilla posible, es muy fácil hacer mucho más difícil esa tarea de lo que es. En ese caso es muy fácil que trabajar con un enfoque visual confunda más que aclare. O desde luego, habrá que aprender a trabajar con él y acostumbrarse igual que con el enfoque lógico. Aunque teóricamente tengamos más familiaridad con el enfoque visual.

El enfoque lógico también facilita extraordinariamente las tareas sencillas. Pongo un ejemplo básico. Cada vez es más común usar en ámbito profesional documentos que estén firmados electrónicamente. También es más corriente que haya que anexar documentos a otros documentos electrónicos. PDF es el formato perfecto para esa finalidad. Es muchísimo más fácil crear un documento con una lista de documentos adjuntos desde un enfoque lógico, que tener que adjuntar y hacer visualmente un listado de archivos adjuntos en el documento principal[10].

Dentro del enfoque visual, supongo el programa más parecido a ConTEXt sería *Adobe InDesign*. Reconozco que nunca he usado este programa, por lo que puedo comprarar ambos, pero sin aludir a la experiencia de uso el segundo programa[11]. No sé si uno es mejor que otro, aunque *Adobe InDesign* tiene las limitaciones del enfoque visual. Al ser un programa bastante más complejo que un procesador de textos, el enfoque visual puede llegar a poner la tarea muy complicada.

*Adobe InDesign* requiere también aprendizaje. Es necesario que la persona que lo use se adapte a su uso y adecúe su tarea al programa. En eso estaría en igualdad de condiciones con ConTEXt. Pero hay en

algo que *Adobe InDesign* no puede competir con ConTEXt, además de la mayor maleabilidad lógica. Es en su coste. ConTEXt se ofrece con licencia *GNU General Public License* versión 2[12], que permite su libre distribución de versiones íntegras o modificadas. Se ofrece gratuitamente para quien lo quiera usar. Además, se actualiza constantemente. Evidentemente, la gratuidad es una virtud para algo que es bueno. ConTEXt está desarrollado por profesionales para los que constituye su herramienta de trabajo. Este libro trata de mostrar por qué ConTEXt es una grandísima herramienta.

# TIPOGRAFÍA DIGITAL

*Corrección tipográfica   El arte de la tipografía   ¿Qué es tipografía digital?*
*Composición automatizada   Complicación y complejidad*

## CORRECCIÓN TIPOGRÁFICA

Si atendemos a su origen, podríamos decir que la tipografía es la escritura con tipos. Eso no tendría más problema, si no hubiese dos modos de esa escritura: la mecanografía y la imprenta. En inglés nos encontramos tanto con *typewriting*, como con *typesetting*. La primera es la mecanografía —la escritura con tipos metálicos—. La segunda es la tipografía de imprenta —la composición con tipos metálicos—. La mecanografía no permite ninguna movilidad de los tipos, mientras que la imprenta exige esa movilidad. Como ejemplo, la práctica —curiosa, o tipográficamente nefasta— de entregar borradores sin justificación de margen a la derecha tiene sentido con la máquina de escribir. Porque en ese caso es imposible conseguirla. Sin embargo, es desacertado con caracteres de ancho variable, como veremos más adelante.

Cuando escribimos con ordenadores, aumentan nuestras posibilidades de precisión del texto. Las comillas deben ser las propias, que en el caso del español son las comillas «latinas» —incluso las ‹simples›—. Aunque su uso está generalizado en nuestro idioma, las denominadas comillas "tipográficas" son propias del inglés [13]. Otros ejemplos de lo que se puede llamar ortotipografía —o incluso ortografía tipográfica—, son el uso de cursivas para títulos de obras o expresiones en lenguas extranjeras respecto al propio texto. Aunque no hay que poner en cursiva las expresiones extrajeras escritas en alfabetos distintos al propio del texto [14]. También es importante añadir las citas siempre entrecomi-

lladas, para saber qué parte del texto es de quien escribe y qué parte toma prestada [15]. Otros ejemplos serían el uso de rayas —para añadir frases— o guiones de intervalo: 20–50. El uso adecuado de las letras extranjeras es importante. El nombre propio danés Søren, no Sören o Soeren. Tampoco es *Strasse*, ni *Straβe* o *StraBe*, sino *Straße* —«calle», en alemán [16]—.

También las ciencias tienen su escritura correcta. Son abreviaturas de unidades de medida: $cm^2$ y $m^3$. La formulación química del agua es $H_2O$, o $CO_2$ es el dióxido de carbono. El signo negativo no es un guión: −2. Ni la multiplicación es una *x*: 3 × 2. Los ejemplos pueden ser infinitos. Sólo se trata de usar el carácter adecuado.

La corrección tipográfica no es tipografía. En todo caso, la da por supuesta. La tipografía es el arte de la imprenta. Tiene que ver con la organización del espacio de texto dentro de la página. Para que la imprenta lo saque en papel. Quienes escribimos con procesadores de texto, organizar el texto en el blanco de la página nos puede sonar raro. Porque casi no hay más que poner márgenes y elegir tipo de letra. La composición tipográfica es algo distinto, como trataré de mostrar en el siguiente epígrafe.

## EL ARTE DE LA TIPOGRAFÍA

La tipografía es la técnica —el arte entendido como «saber harcer», como τέχνη— de componer textos impresos. Es la posición conjunta de tipos móviles metálicos que conforman el texto. Impregnada con tinta, esa composición metálica se presiona contra el papel, generando así el texto impreso. De este modo, la tipografía es el arte de saber cómo hacer esa composición.

La tipografía tiene que hacer máximamente legible el texto. La legibilidad que se consigue con la imprenta es mayor que la que se obtiene con caligrafía o mecanografía. Se componen fundamentalmente libros.

Como la extensión del texto de un libro ocupa varias páginas, es necesario que su lectura esté especialmente facilitada desde el punto de vista tipográfico. No sólo son los párrafos de una novela, pueden ser tablas o diferentes cuerpos de notas y anotaciones marginales. Se trata de mostrar el texto de la manera en que sea más fácil de leer.

La composición tipográfica no es diseño gráfico. Con la introducción de la informática, esta diferenciación puede parecer menos clara. La composición del texto no es propiamente diseño, porque su exigencia de legibilidad no responde a reglas sujetas a modas. La legibilidad de un texto es distinta de la apariencia gráfica que ha de tener una revista —ilustrada, no publicación periódica—. En un libro, el texto es el elemento central, aunque pueda estar acompañado de imágenes. En una revista, el texto tiene un papel secundario respecto al protagonismo de la imagen. Éste es el motivo de que la legibilidad es esencial para el texto impreso. Su belleza es secundaria —o más sobria— comparada con el diseño gráfico.

La tipografía consiste en ese componer un texto en las páginas que necesite. No es simplemente meterlo en páginas sin más. Requiere observar el modo en que se consigue la máxima legibilidad atendiendo a qué cuenta el texto. Es una labor de observación y detallle. La composición tipográfica da forma visual al texto. De ahí que la tarea tipográfica deba ser invisible. Lo único que ha de hacer es invitar a la lectura, siendo lo más transparente posible. Algo falla si la composición tipográfica irrumpe en la lectura de lo escrito. Como la elegancia y las buenas formas, no debe notarse, aunque siempre haya de estar presente.

### ¿Qué es tipografía digital?

La tipografía digital es la composición tipográfica del texto con medios informáticos. No se arma un texto con tipos de metal, para

luego entintarlo e imprimirlo en una hoja de papel. Los tipos digitales se usan para generar un archivo informático en el que están colocadas las letras en una página virtual. Los tipos no son trozos de metal, la página no es de papel y no hay tinta que manche. Pero la tipografía digital trata de ordenar marcas negras sobre fondo blanco. Ésta es la tarea propia de la composición tipográfica [17]. Esas marcas negras pueden proceder de tipos *OpenType* y su conjunción con el fondo blanco puede ser un archivo PDF [18].

Sin embargo, no todo uso de un ordenador para imprimir textos es por sí mismo una tarea de composición tipográfica digital. Aunque se usen los mismos elementos. Es cierto que un procesador de textos realiza una determinada composición de página. Pero tiene unos límites que hacen imposible una auténtica tarea de composición tipográfica. En este sentido, el procesador de textos informático constituye la evolución de la máquina de escribir eléctrica con memoria. Muchos usos del procesador de textos podrían considerarse propios de una máquina de escribir electrónica.

El procesador de textos no sigue la línea de evolución de la imprenta, porque no realiza labor de cajista digital —de composición tipográfica—. No se trata de un defecto de un procesador de texto, como programa informático no puede ser un tipógrafo digital. Los tipos digitales incorporan cada vez más propiedades tipográficas que pueden aplicarse también en procesadores de texto. Pero hay cuestiones tipográficas que están por encima del nivel de la letra y que no son propias de programas que no sean específicos de composición tipográfica. Como ya he dicho, no es que un procesador de textos sea peor, sencillamente es una herramienta distinta. Un programa de composición tipográfica tiene mucha más precisión, requerida por la tarea que tiene que realizar.

La tipografía de imprenta exigía que la composición con tipos metálicos fuese manual. La tipografía digital permite que esa composición

sea mediante órdenes informáticas. La colocación de las letras y de las unidades superiores que forman no se puede hacer a mano en un archivo digital. Es necesario hacerlo mediante órdenes informáticas. Independientemente de que haya que escribir el código directamente —como en ConTEXt—, o sea mediante una simulación visual —como con *Adobe InDesign*—, tenemos que especificar al programa adecuado cómo queremos que trate cada una de las letras y los conjuntos de letras que puedan formarse —palabras, líneas, párrafos…—. La tipografía digital parte de dos elementos: el texto y el formato visual que queremos dar al texto. La tarea de composición tipográfica supone usar el programa para que aplique el formato visual al texto deseado, para que coloque cada cosa en su sitio. De esa ordenación surge el texto compuesto tipográficamente.

La mayor precisión de la composición tipográfica supone un mayor potencial para el detalle. Si bien necesita de conocimientos especializados, la composición tipográfica de un texto está al alcance de cualquiera. Porque no se necesita una imprenta para llevarla a cabo. Y porque puede imprimirse con una calidad relativamente alta sin tener que usar una imprenta. La informática permite tener la funcionalidad de la imprenta con los medios de la máquina de escribir. Eso quiere decir que el cuidado propio por el detalle que necesita el texto impreso puede conseguirse empleando muchos menos medios que si tuviésemos que editar un libro de gran tirada.

Pongo un ejemplo propio de tipografía digital, que por desgracia he tenido que contemplar en dos libros —de una prestigiosa editorial española y de un conglomerado editorial mundial—. A la izquierda está lo que me he encontrado impreso. A la derecha está lo que la tipografía exige.

filósofo flemático          filósofo flemático

*finura afligida* *finura afligida*

Es sólo un detalle. Es obvio que la tipografía no es sólo la ligadura de letras —como *fi* o *fl*[19]—. Aunque se trata de una omisión especialmente grave en empresas que venden libros[20]. No se trata aquí del dicho alemán *der Teufel steckt im Detail* —«el diablo se esconde en el detalle»—. En esos detalles como los que he mostrado el ejemplo se manifiesta una calidad —o se rebaja—. El trabajo del detalle no es para que alguien lo vea, sino para que esté bien hecho. Es como elegancia, la auténtica es el propio modo de ser.

El ejemplo de las ligaduras también muestra otro aspecto que he mencionado. Como la ligadura de letras es algo que puede definirse en el tipo digital —en formato *OpenType*—, esta propiedad puede estar disponible ahora en procesadores de texto. En TeX, estaba ya disponible hace casi cuarenta años.

Del mismo modo que la imprenta supone una mayor calidad respecto a la mecanografía, esa mayor precisión permite que la composición tipográfica digital alcanzar mejores resultados que el proceso informático de textos. Como la tarea es más completa, es más complejo manejar ConTeXt que un procesador de textos. Del mismo modo, la imprenta era más complicada que la máquina de escribir. La complejidad se puede traducir en dificultad de aprendizaje. Aunque tenemos que valorar qué perseguimos con el uso de una herramienta para ver si se adecúa a nuestros fines. Así podremos valorar si tiene sentido que aprendamos algo nuevo.

## Composición automatizada

La tipografía digital convierte al ordenador en un gran cajista digital. De un modo que intuyo que *Adobe InDesign* es incapaz, ConTeXt distribuye los diferentes elementos del texto según las instrucciones

que le damos. Esas instrucciones son únicamente especificaciones. Algunos ejemplos serían: las notas al pie forman un párrafo en cada página, al comienzo de cada capítulo hay un sumario de las secciones que contiene, o hay que generar un vocabulario completo con los contenidos de cada página [21]. No tenemos que tomar cada parte del texto y ponerla en su sitio. La composición tipográfica automatizada es precisamente que esa tarea la haga el ordenador.

En el fondo, podría decirse que estamos entablando un diálogo con el ordenador. Le advertimos que el texto que le damos tiene que estar de una determinada manera. Por supuesto, el ordenador no adivina nada. Tenemos que decirle qué es cada elemento del texto, por ejemplo, qué es un título o qué una nota al pie. También hemos de aclararle cómo tiene que aparecer todo en la presentación final. Le ponemos a trabajar, como diciendo: «ahí te las compongas» [22]. ConTEXt compone de esa manera el texto.

No tenemos que colocar nada, porque es el programa el que tiene que colocar todo. Para eso sólo tenemos que saber decirle cómo tiene que hacer todo. El ordenador es nuestro asistente digital con capacidades tipográficas plenas. Manejado correctamente, es muy fácil que usarlo para las tareas complicadas sea mucho más rápido que un sistema visual. Con un ejemplo básico, una nota al pie. ¿De verdad creemos que es más rápido el diálogo visual de un procesador de textos para insertar una nota al pie que teclear `\footnote{}` [23]?

Con el ejemplo de las notas podemos ir más allá. En una traducción, podríamos necesitar añadir notas a la traducción. Tendrían que ser distintas del texto normal. La numeración con letras, números romanos, asteriscos u otros símbolos es una opción. Intentar que estén aparte de las notas del texto —tanto al pie, como al final— es ya más complicado. Composición automatizada es todo eso y bastante más. Sólo tendríamos que escribir las notas de cada clase, en el punto del texto

al que se refieren. ConTEXt ya se encargará de colocarlas donde han de estar situadas. Por supuesto, numerando cada nota correctamente.

### COMPLICACIÓN Y COMPLEJIDAD

Este par de conceptos pueden ayudarnos a tratar de esclarecer la mayoría de incomprensiones que pueden suscitarse en torno a ConTEXt. Complejo es lo contrario de simple. Complicado es lo opuesto a sencillo. Dado que simple es lo que carece de partes, complejo es lo que está compuesto de diferentes elementos. Complicado es lo que está enmarañado de tal modo que hace difícil que pueda entenderse fácilmente. Complejo y complicado no son necesariamente sinónimos, aunque a veces los usemos como tales. Lo complejo puede ser relativamente sencillo y podemos complicar también lo que es simple.

Es lógico que las herramientas informáticas tengan diferentes elementos. Estos elementos se pueden presentar de manera sencilla o de manera complicada. No sólo es cuestión de informática, lo podemos advertir en otros ámbitos —en los que todavía podamos pensar que no todo se reduce a informática—. Compleja o simple es habitualmente la tarea, no tanto la herramienta. Atornillar un tornillo cualquiera es una tarea simple y un destornillador es una herramienta sencilla. Ahora, pretender pilotar completamente un avión con un único botón pulsador sería hacer imposible una tarea que es esencialmente compleja. Aunque quienes sepan pilotar y tengan muchas horas de vuelo puedan afirmar que es algo sencillo. Si fuese absolutamente obvio, literalmente cualquier persona podría pilotar un avión. Sólo es sencillo después haber aprendido.

Las herramientas informáticas también son más simples o complejas según la tarea que tengamos que realizar con ellas. Grabar de un micrófono con un ordenador puede ser algo sencillo o increíblemente complejo. No es lo mismo estar hablando con alguien por cualquier

sistema de mensajería instantánea que editar electrónicamente sonido para una grabación de un disco. En ambos casos podemos decir que se trata de grabar sonido con el ordenador. Cuanto más precisa haya de ser la herramienta informática, necesariamente más compleja habrá de ser. Más complicado puede resultarnos su uso.

Es posible que las herramientas nos resulten complicadas porque sean demasiado complejas para la tarea que queremos realizar con ellas. Es fácil que en informática nos pase eso. Por seguir con el ejemplo anterior, es fácil que nos resulte innecesariamente difícil grabar un archivo de sonido con nuestra voz usando un programa de edición de sonido digital propio de un estudio de sonido. Porque estamos matando moscas a cañoñazos.

Si no tenemos interés en absoluto en ser capaces de generar documentos con buena calidad tipográfica o un procesador de textos decente satisface perfectamente todas nuestras necesidades, es muy posible que ConTEXt no nos aporte nada. Por el contrario, si necesitamos crear documentos empleando un enfoque fundamentalmente lógico o necesitamos buena calidad tipográfica, entonces ConTEXt puede ser la herramienta perfecta. Es posible que sea interesante para quienes usen *Adobe InDesign* como programa para creación de documentos electrónicos.

No se trata de hacer las cosas innecesariamente complicadas. Ahora, tratar de simplificar lo complejo supone a menudo hacer imposible o extraordinamiente difícil una tarea. Con otro ejemplo, pretender tocar *Sultans of Swing*[24] sin aprender un acorde es ingenuo. O como decía un método de enseñanza de idiomas: «el alemán sin esfuerzo». Me temo que tal cosa no existe, por mucho que *Assimil* publicite y trate de vender lo contrario. El aprendizaje exige esfuerzo, aunque no necesariamente sufrimiento. Todo el esfuerzo reside entender cómo lo que es complejo y nos parece complicado, es sencillo en el fondo.

# EL APRENDIZAJE Y LA PRÁCTICA

*¿Para quién es este libro?* *Frustración informática* *Sobre el aprendizaje* *Sobre la práctica* *Es sólo cuestión de práctica*

## ¿PARA QUIÉN ES ESTE LIBRO?

Estas páginas serán útiles a quienes tengan interés en aprender ConTEXt. Esto supone que tengan un interés real en introducirse en el campo de la tipografía digital. No hace falta conocimientos previos de código para formato de texto, como XML o incluso LaTeX —si no te suenan, no pasa absolutamente nada—. Podrán facilitarnos el aprendizaje, pero no son imprescindibles.

Tampoco es necesario es necesario tener conocimientos de inglés para seguir este libro. Ahora bien, saber algo de inglés nos facilitará mucho la escritura y la comprensión del código en ConTEXt. Además, es importante señalar que existe muy poca información en español sobre ConTEXt. Si el inglés nos resulta difícil, tendremos un mayor obstáculo para seguir aprendiendo. Entiendo que con el nivel medio que se enseña en España como lengua extranjera debe ser suficiente.

Quien quiera aprender debe tener cierta experiencia de uso de ordenadores. Es necesario saber instalar programas o estar dispuesto a hacerlo. Se necesita también saber abrir archivos, escribir en el teclado y grabar archivos —como en un procesador de textos—. Y en internet, saber usar un navegador y poder buscar información. Para el uso más avanzado, para informar de incidencias con el libro, es necesario saber manejarse en *GitHub*, *Bitbucket* o *GitLab*, abriendo y respondiendo a incidencias en esos sistemas.

El libro no supone experiencia previa con ConTEXt, TEX o código alguno de etiquetado de textos. En la medida en que sea posible, trataré de explicar todo. No obstante, el libro y las explicaciones serán progresivas, lo más básico se explica al principio y lo ya explicado no se vuelve a explicar. En caso de necesitarlo, tendrás que volver sobre la explicación original.

Dicho esto, es importante advertir que, como con todo sistema informático novedoso, el aprendizaje requiere tiempo. Es posible que con diez horas ya podamos hacer documentos sencillos con ConTEXt. Pero para tener una cierta idea de uso autónomo, probablemente necesitemos bastantes más horas. Por eso, al igual que necesitaríamos con otro sistema parecido, necesitamos paciencia y dedicarle tiempo a la tarea de aprender.

### frustración informática

Para que nadie se lleve a engaño, me parece importante advertir de una situación bastante común en informática. Los ordenadores son máquinas que siguen instrucciones que las personas desconocemos muy a menudo. Podemos pensar que estamos haciendo algo bien y que el ordenador no funciona, cuando en realidad no sabemos hacer eso que pensamos para que el ordenador lo lleve a cabo. Si el tiempo apremia, o hemos dedicado demasiado a una tarea sencilla, eso suele provocarnos un hondo sentimiento de frustración. Le pasa a todo el mundo, porque nadie nace sabiendo. Que algunas personas tengan mayor manejo informático, muchas veces sólo significa que sus límites están más avanzados que los del resto.

El manejo de ordenadores nos puede exasperar en algún momento. Es algo que hemos de tener muy en cuenta cuando nos proponemos aprender el manejo de un programa informático. Puede haber momentos en que necesitemos que algo funcione y no lo haga como debiera

—o como nos parece que debería ser—. El aprendizaje de ConTEXt también supone saber enfrentarse a esas situaciones. Muchas veces nos gustará soltar toda la exasperación que llevamos dentro.

Es necesario tener paciencia para poder aprender. En este caso, para poder analizar qué falla, poder buscar información y preguntar si fuese necesario. Nadie pide que tengamos nervios de acero. Sin embargo, si mantener la calma —al menos después de la tempestad— nos supone gran esfuerzo, debemos considerar si nos interesa realmente aprender ConTEXt. Si no nos encontramos en disposición de aprender también de nuestros propios errores, quizá deberíamos replantearnos este aprendizaje.

Por si sirve de medida, me he dado de bruces varias veces contra errores tontos en ConTEXt. Su único causante y responsable he sido yo mismo. He sentido ganas de estrellar el portátil contra la pared, por muy breves momentos. Espero haber aprendido a mantener un poco más la calma, a evitar llegar a la tempestad. Porque si hay un problema, cuanto antes empecemos a buscar la solución, antes los solucionaremos. Con la exasperación, nadie puede llegar a aprender. Ni más, ni mejor.

Como última advertencia contra la frustración, es importante que no olvidemos lo siguiente. Nadie tiene que sufrir porque estemos aprendiendo ConTEXt. Puede que el proceso de aprendizaje nos pueda llevar al límite de nuestras capacidades. Normalmente, no será para tanto, aunque podamos pensarlo. Por falta de conocimientos, podemos pensar que ConTEXt no está haciendo lo que debe. Depediendo de la presión que nos metamos, puede llegar a ser algo realmente exasperante. Nuestro entorno personal —o profesional, si fuera el caso— no tiene que notar siquiera nuestro enfado. Para poder ver las cosas correctamente, necesitamos despejar la cabeza, distraernos. Sólo así

podríamos pedir ayuda cuando la necesitemos, como el favor que es. Sin exigencias y sabiendo esperar el tiempo que lleve.

### SOBRE EL APRENDIZAJE

La parte más complicada de toda indicación para el aprendizaje estriba en hacerse cargo de qué es comprensible y qué no es comprensible para quien trata de aprender. Uno de los campos en donde se muestra más claramente esa necesidad de hacerse cargo con lo que la persona entiende es la informática.

El aprendizaje también supone ir ganando cada vez una mayor comprensión de lo que vamos aprendiendo. Es convertir progresivamente lo desconocido hasta entonces en conocido. Lo que nos parece imposible de entender, poco a poco nos va mostrando su sentido. Por eso, lo importante no es siempre entender todo perfectamente a la primera. Demasiadas veces creemos que sabemos ya cosas de las que después nos damos cuenta de que no las habíamos entendido en absoluto. A menudo, tendremos que volver a revisar lo que ya creíamos sabido, para afianzar nuestros conocimientos.

Siempre cabe entender más y mejor. Por eso, es posible que haya cosas sobre las que tengamos que volver para poder entenderlas. No me refiero a memorizarlas, a saberlas recitar mecánicamente. Me refiero a saber qué es lo que estamos haciendo —o qué hace ConTEXt— en un determinado momento. En caso de que no lo entendamos tras repetidos esfuerzos, será lógico preguntar.

El aprendizaje es siempre personal. Nos pueden enseñar, pero es siempre cada cual quien aprende. Aunque parezca obvio, nadie puede aprender por nosotros. En ese sentido, es cierto que todos somos autodidactas. Además, en este caso, nos enseña un libro. Eso quiere decir que este libro representa un compromiso de cada cual con su aprendizaje. Ese compromiso expresa madurez mental.

No va a haber nadie detrás obligándonos a aprender. La madurez mental supone no desanimarse a la primera dificultad. Está garantizado que encontraremos dificultades. Si no las encontrásemos, sabríamos ya manejar ConTEXt. La decisión madura es seguir buscando cuando parece que no podemos seguir, que se ha acabado. Buscar puede ser volver sobre lo ya leído, tratar de encontrar información en internet, o preguntar en el lugar adecuado. La paciencia supone también saber esperar a que encontremos la solución. No siempre será inmediata.

## SOBRE LA PRÁCTICA

Toda la utilidad que puede tener este libro se dirige a poder aprender una técnica de composición tipográfica de textos. Por tanto, no se trata de una consideración teórica sobre la tipografía —me faltarían conocimientos para eso—. Es una introducción a una técnica, que al mismo tiempo es un arte. Eso es lo que en la antigua Grecia se llamaba τέχνη, la raíz de TEX que se encuentra en ConTEXt.

El libro es lo que tenemos que leer para aprender. Aprender en este caso no es necesariamente saberse todas las órdenes de memoria. Recordaremos las órdenes por las veces que las usemos, por eso no debemos preocuparnos. Si necesitamos consultarlas, tendremos este libro y otras fuentes de consulta. El objetivo al leer cada capítulo es aprender mediante la práctica. Esto supone leer la explicación detallada con atención y practicar con el código que se ofrece. Sin esta práctica, este libro es sólo una pérdida de tiempo para quien lo lea. En el fondo, todas estas páginas están estructuradas como un gran ejercicio de ConTEXt.

Practicar con el código requiere que tecleemos todos los ejemplos que se ofrecen[25]. Éste no es un libro que contenga ejercicios, sólo contiene una explicación introductoria de cómo funciona ConTEXt. Ese funcionamiento se muestra con ejemplos de código. Cada persona tie-

ne que teclear el código para ver cómo funciona. Eso no sólo supone copiar literalmente los fragmentos de código, sino probar con todas las opciones que se dan. Se necesita probar a cambiar y ver qué pasa. Es la única manera posible de aprender. Al menos, con este libro[26].

Una última cuestión respecto al aprendizaje por medio de la práctica. El libro no ofrece los resultados del código que se incluye. Es sencillamente una invitación —más o menos forzada— a probar el código. En principio, el código está libre de errores[27]. Tener que ejecutar el código para conseguir resultados es otro modo de implicar a quienes leen en su proceso personal de aprendizaje.

### es sólo cuestión de práctica

Debido a la materia que trata —el uso de un programa informático en el que hay que escribir código— y al modo de aprendizaje —mediante un libro—, debe quedarnos muy claro que éste no es un libro para leer sin más. Lo sería tanto como es una partitura musical o un libro de recetas de cocina. Por supuesto que cabe sólo leer partituras o recetas de cocina, pero ambas tienen sentido —algunos dirían que sólo existen— en su ejecución.

Tocar un instrumento musical como el piano, supone poner los dedos sobre las teclas y presionar sobre ellas. Hasta a un absoluto ignorante musical como yo, se da cuenta de que es el mínimo para poder empezar a aprender. Del mismo modo, no aprenderemos nunca a cocinar si no nos manchamos las manos con los ingredientes de las recetas. De hecho, también tendremos que probar cómo nos van saliendo los platos.

ConTEXt no es diferente en este punto de los dos ejemplos de música y cocina. Si el código —los fragmentos escritos a máquina de escribir y con tonos de gris— nos limitamos a leerlo, es casi imposible que aprendamos nada en absoluto. Sobre todo, porque somos las personas

las que escribimos el código. Es ConTEXt quien se encarga de leerlo y ofrecer el resultado de las órdenes que se le dan. Cada persona debe aprender a dar órdenes al programa. No hay otra manera de llegar a saber que haciéndolo. Debemos tomar todo ejemplo de código como un ejercicio que debemos practicar. Aunque sólo sea para ver cuál es su resultado. Para verlo en pantalla, no para leerlo en este libro.

El documento PDF en su versión electrónica ofrece anotaciones con texto del código referido a la instalación de *ConTEXt Suite*. No se ofrece para nada más. Hay dos razones para esa decisión. La primera razón es que es imposible aprender órdenes —no de memoria para recitarlas, sino cómo se escriben— si no las tecleamos. La única manera de familiarizarse con esas órdenes es mediante la escritura repetida. La segunda razón para no ofrecer anotaciones del código de ConTEXt es que sólo serviría si los fragmentos que copiásemos siempre los fuésemos a usar así siempre. La capacidad de las órdenes es la posibilidad de mezclarlas para conseguir resultados diferentes. El motivo de escribir siempre las órdenes es aprender a manejarlas, no el ejercicio mecanográfico.

Quizá toda la valoración de si este libro —probablemente también si esta tarea de aprendizaje— puede llegar a cumplir nuestras expectativas, depende de lo siguiente. O tenemos la determinación de practicar con cada fragmento de código, o la simple lectura nos parecerá una tortura. Porque además de que no veremos ningún resultado, todo nos parecerá increíblemente abstruso y no aprenderemos —casi— nada. Por supuesto, la decisión tiene que tomarla cada cual. Sólo espero haber ofrecido elementos de juicio suficientes para estar en situación de tomar una decisión.

# CONVENCIONES TIPOGRÁFICAS USADAS

*Documento para impresión en papel* *Documento para dispositivos electrónicos*

Para facilitar la lectura de este manual se ofrecen dos versiones. Son dos formatos de archivo o dos tipos de documento: PDF y ePub. El primero está pensado para su impresión en papel. El segundo está pensado para su visualización en lectores de tinta electrónica.

El documento ePub no permite tanta capacidades tipográficas como el formato PDF. Por tanto, trataré de que no se pierda ninguna información esencial, si bien existen limitaciones como explico en *Edición en formato ePub*.

## DOCUMENTO PARA IMPRESIÓN EN PAPEL

Como norma general, este texto usará en la mayor parte de su extensión la tipografía romana *TEX Gyre Pagella*. Sólo a modo de ejemplo, se usarán ocasionalmente otras tipografías romanas. Las letras del alfabeto griego se completan con la tipografía *GFS Didot*.

El tipo *cursiva* se usa para marcar énfasis, expresiones extranjeras o productos. Respecto a las marcas registradas, la referencia será únicamente identificativa.

El tipo VERSALITAS se emplea como variante estética en los títulos. Podrá también usarse ocasionalmente para señalar la autoría en una referencia bibliográfica.

El tipo negrita no se emplea en este libro porque considero que produce una excesiva recarga visual, que puede distraer la lectura. Para favorecer la lectura del código, los fragmentos del código emplean tonos diferentes en escala de grises.

La tipografía `mecanografiada` de la familia *Cousine* se emplea para todos aquellos textos que se introducen mediante el teclado. Pueden tratarse de código —ya sean fragmentos, órdenes u opciones aisladas—, algunos programas o direcciones de internet.

Las expresiones de código dentro de un párrafo de texto pueden contener un guión bajo la vocal para partir la palabra al final de la línea en que se encuentre. Este uso exclusivo del guión bajo será la única separación de palabra para el código. Así se hacía con las máquinas de escribir tradicionales. También se indica que no hay que teclearlos. Sin embargo, los guiones normales o bajos sin carácter encima, como en `first-setup.sh` o `PDF32000_2008.pdf`, hay que teclearlos, puesto que forman parte de la expresión que se trate.

La tipografía redonda Optima se usa únicamente para señalar diálogos de entornos de ventanas en algunos sistemas operativos.

### Documento para dispositivos electrónicos

*Tempestiva* es la tipografía usada para documentos ePub. Contiene caracteres para los alfabetos latino, griego y cirílico.

Debida a la cantidad de código contenida en estas páginas, se emplea también `Cousine` como tipografía mecanográfica. Ambas tipografías se incluyen en el documento ePub para facilitar la lectura.

# EDICIÓN EN FORMATO EPUB

*Diferentes formatos    Diferentes finalidades    Creación de documentos    El resultado final*

Este libro trata de composición tipográfica digital. Aunque puedan estar muy relacionadas —o incluso compartir campos comunes—, no versa sobre edición digital[28]. Estas páginas son una explicación de cómo se edita en ese papel digital que es el documento PDF. El formato principal de *Aprender ConTEXt* es PDF. Sin embargo, no puedo dejar de ofrecer un documento en un formato para dispositivos digitales. El estándar para lectura de documentos en dispositivos digitales —en ordenadores con diferentes tamaños de pantalla— es el formato ePub.

Quizá te sorprenda mi afirmación anterior, porque PDF parece ser el formato de intercambio de archivos que no se modifican. Realmente, PDF es un formato industrial de impresión[29]. Es excelente en su tarea. Dentro de esa definición del papel digital que contiene, lo podemos usar para intercambiar documentos con idéntico formato. Como todo, nos encontraremos limitaciones cuando le asignamos una tarea distinta de aquélla para la que está diseñado. La potencia de la página digital para la impresión es un límite en la visualización en diferentes dispositivos.

## DIFERENTES FORMATOS

PDF es una de las materias primas para un tipógrafo digital. Es el papel de un gran cajista digital como ConTEXt. Sin embargo, sin una hoja —física o lógica— de dimensiones determinadas, la composición acabada del texto se vuelve una pesada carga. No es retórica, trata de ver un documento PDF de tamaño de hoja DIN A4 o carta en un

dispositivo no mayor de seis pulgadas. En el mejor de los casos, podrá llegar a leerse, pero no dejará de ser muy incómodo.

El formato ePub contiene una estructuración lógica de los elementos del texto. A ese texto estructurado, le añade un formato de presentación. A diferencia del documento PDF, un archivo ePub no está totalmente «cocinado». Contiene una receta, que cada dispositivo ha de terminar de preparar. Ésa es la razón de que un documento PDF siempre pueda tener un mayor nivel de determinación, de definición. Por el contrario, el documento ePub posee una mayor flexibilidad para adaptarse a la pantalla.

### DIFERENTES FINALIDADES

La idea básica sobre los dos formatos se puede expresar con su capacidad y su limitación. El documento PDF es perfecto para imprimir, pero no lo es necesariamente para leer en pantalla. El documento ePub es pefecto para leer en pantalla y no es buena idea imprimirlo. De modo general —no sólo por *Aprender ConTEXt*—, los diferentes formatos se complementan para diferentes finalidades.

No existe la solución perfecta para todo, al menos en informática. Si lo examinas detenidamente, una hoja de papel y una pantalla de ordenador son medios absolutamente heterogéneos. Incluso las pantallas de diferentes dispositivos digitales no son comparables. De ahí que sea lógico que el formato único para documentos no exista.

El documento PDF está pensado para una presentación fija. Por excelencia, es la página impresa. El documento ePub sirve para pantallas menores, en las que el texto fijado es ilegible.

### CREACIÓN DE DOCUMENTOS

La generación de documentos electrónicos está determinada en tres niveles:

- El formato del documento.
- El programa que genera el documento.
- El dispositivo y el programa que visualizan el documento.

Como he dicho, estos tres niveles se aplican a toda creación de documentos. Puesto que implica limitaciones, abordaré de modo esquemático la generación de documentos ePub.

### EL FORMATO

El formato ePub segunda versión tiene ciertas limitaciones comparado con PDF. Quizá la más básica es la falta de numeración de títulos[30]. Puede llegar a suplirse la carencia, si el programa de creación de documentos ePub añade los números automáticamente.

En la práctica no existe partición silábica de palabras al final de la línea. Aunque pueda haber dispositivos que la añadan[31], no es algo generalizado en los lectores de tinta electrónica. Sobre todo, no tienen partición silábica en todos los idiomas. Existe un modo de añadir esa información, aunque tiene sus limitaciones.

### LA HERRAMIENTA

`pandoc` es el programa que uso para generar la versión ePub de *Aprender ConTEXt*[32]. En la fecha de la redacción de este texto, la versión 1.19.2.1 de `pandoc` tiene las siguientes limitaciones que afectan al resultado del documento ePub:

- Es capaz de añadir números a los títulos. Sin embargo, tiene que ser a todos los títulos, la numeración es jerarquizada y la numeración no puede ser distinta de la aritmética —no cabe usar letras—.

- No puede añadir el índice o sumario en el contenido del texto del documento ePub. En realidad, sólo puede añadirlo en una posición fija. Me temo que si no puedo colocarlo, no me interesa tenerlo como parte del texto.

- No puede colocar todas las notas en un mismo capítulo, de modo que su numeración sea corrida. De este modo, las notas reinician su numeración en cada parte en el documento ePub. El documento PDF contiene notas con numeración corrida.

### EL DISPOSITIVO DE LECTURA

El programa de la mayoría de lectores de tinta electrónica —excluido *Amazon*, que no sigue el estándar ePub— es *Adobe Reader Mobile*. Dependiendo de las versiones, sus limitaciones serán mayores o menores. Al editar un texto, hemos de tomar las capacidades del mínimo común denominador. De otro modo, podemos causar problemas de visualización graves, como alguno de los que describiré. La razón de que el patrón de edición sea el mínimo común denominador de los lectores electrónicos es muy simple: no se actualiza. No es ni fácil, ni difícil de actualizar. Aunque parezca increíble, un lector de tinta electrónica no tiene actualizaciones.

Gracias a la generosidad de un amigo, tengo un lector de tinta electrónica *Sony Reader PRS–T1*. Encontré muchas limitaciones, de las que contaré las más graves. Por casualidad, me topé virtualmente con uno de los ingenieros de *Adobe* en una lista de distribución de correo electrónico. Uno de los problemas era que *Adobe Reader Mobile* no era capaz de abrir un documento correcto. Me puse en contacto con él y le dije si podrían corregir el fallo. Era algo muy básico y lo corrigieron. El problema era actualizar el programa, como el propio ingeniero de *Adobe* reconoció. Existen muchos más fallos, pero no tiene sentido informar sobre ellos. Porque para disfrutar de la versión corregida,

tienes que estar seguro de que la que usarás se publica antes de que se fabrique el lector.

Te puede parecer raro que no pueda actualizarse el visor de documentos ePub de un lector de tinta electrónica. Por extraño que suene, no se puede. Sólo si el fabricante del lector —*Sony* en mi caso— compra la nueva versión, puede hacer una actualización para el dispositivo lector. En la práctica, no lo hacen. Ni siquiera es posible cambiar las tipografías básicas del dispositivo o añadirle tipos de letra al sistema[33]. ¿Nos parecería una limitación aceptable para un ordenador? Precisamente eso es un dispositivo lector de tinta electrónica.

Un detalle que falla en la versión que uso de *Adobe Reader Mobile* son las versalitas. Aunque eso quizá no sea un problema, porque es simulación de versalitas. Algo mucho más grave es que no funcione la orientación del texto de derecha a izquierda. No me he equivocado, porque así se escriben árabe y hebreo. Que no sean lenguas que escribamos o leamos, no quiere decir que sean lenguas irrelevantes. Es un fallo bastante grave.

Entre los fallos que nos afectan directamente está la ausencia de formato por idioma[34]. Tampoco es posible añadir automáticamente contenido de texto a los elementos[35]. Eso limita ciertas posibilidades de presentación del texto de estas páginas.

La composición tipográfica básica es la del párrafo. Como veremos más adelante[36], una característica esencial para que el párrafo no tenga espacios desiguales —cuando no demasiado grandes— entre palabras es la partición al final de la línea. Hay dos posibilidades de partición: de palabras en diferentes idiomas, y de direcciones de internet o en los nombres de archivo. Aunque te parezca extraño, las palabras de una dirección de internet o de un archivo pueden contener guiones como parte de su nombre. Como `http://www.aprender-context.tk` o `archivo-pruebas.tex`. No pasa nada por partir la línea después del

guión en esos casos. Pero sería nefasto añadir guiones de más, porque no sabríamos si son necesarios al escribir los nombres.

Cada una de las dos particiones descritas se pueden conseguir con dos caracteres especiales. La partición de palabras se consigue añadiendo un guión oculto, que sólo aparece si la palabra tiene que partirse en esa posición. Además de la complejidad técnica de añadir esa información al documento ePub [37], es fácil que el texto se muestre mal. *Adobe Reader Mobile* parte la palabra al final de línea, pero no añade el guión. Es imposible que sirva así, por lo que las palabras no se partirán al final de la línea.

Para partir direcciones de internet o rutas de archivo, se puede lograr en los lugares estratégicos añadiendo un espacio sin anchura. Aunque parezca contradictorio, tipográficamente no lo es. Un espacio sin anchura es lo que permite que una línea pueda acabar en la posición en que se encuentra ese espacio. Como no tiene ancho, si no hay un salto de línea, es invisible en otro lugar. Con un ejemplo, `http://wiki.contextgarden.net`. Podemos añadir espacios sin ancho antes y después de las palabras, así como antes o después de los puntos —eliminando redundancias—. Es perfecto para partir direcciones de internet que el lector pierde fuera de los márgenes. Sin embargo, *Adobe Reader Mobile* muestra ese espacio sin ancho como un espacio de ancho normal que no permite la partición de línea. Es imposible usarlo, aunque haya direcciones o rutas de archivo que se oculten en los márgenes.

## EL RESULTADO FINAL

Como ya he dicho, las versiones PDF y ePub tienen objetivos diferentes. Comparadas tipográficamente, al formato ePub le faltan ciertas propiedades. La más reseñable es la partición de palabras o direcciones de internet al final de la línea. Incluso aunque pueda haber algún dis-

positivo que las añada, el problema será que añada partición silábida a elementos que no tengan que tenerla.

Debido a que el formato ePub no está pensado como un tipógrafo digital, aunque incluya capacidades tipográficas básicas, hay propiedades de este libro que no puede tener. Por ejemplo, es imposible la partición del código dentro del párrafo usando un guión bajo la última letra de la sílaba. Tampoco tiene sentido que incorpore propiedades microtipográficas como la prominencia en la línea o la elasticidad de caracteres. Estas dos últimas características ayudan a una composición de línea más precisa. Aunque estuviese disponible, sería absurdo aplicarla sin partición silábica de las palabras.

Además de esas cuestiones, los títulos en la versión ePub no tendrán numeración alguna. La carencia de numeración en los títulos también afectará a las referencias cruzadas. Las notas al pie no tendrán un número único, como en el documento PDF. Por último, el archivo ePub no tendrá sumario en el texto del documento. De todas formas, tiene los marcadores como en la versión PDF —aunque sin números, como ya he advertido—.

# SOBRE LA LICENCIA DE USO

*Uso no comercial*   *Sin obra derivada*   *Sólo descarga*   *Un pacto de nobleza*   *Comentarios*

El libro *Aprender ConTEXt. Una introducción a la tipografía digital* se ofrece para descarga gratuita en `http://www.aprender-context.tk`. No está bajo una licencia libre de ningún tipo —incluidas las *Creative Commons*—. La descarga sólo autoriza el uso estrictamente individual y sin ánimo de lucro. No se permite la distribución del libro, ni tampoco su modificación.

Como autor de estas páginas, quiero elegir la mejor forma de hacerlas disponibles gratuitamente a todo el mundo. Se trata de que cualquiera pueda aprender ConTEXt. Por su naturaleza, son necesarias las siguientes condiciones.

## USO NO COMERCIAL

La primera condición prohíbe el uso comercial o con ánimo de lucro. La explicación es muy sencilla. Se trata de un libro que no creo que nunca publique o que venda. No es un compromiso o una promesa. Sin embargo, no quiero que otros lo vendan. Sobre todo, no quiero ser parte del escaparate —aunque sea una parte ínfima— de grandes empresas.

Además de una cuestión práctica explicada en *Sólo descarga*, el uso sin ánimo de lucro no permitiría ponerlo a disposición del público en la mayoría de sitios de internet. No podría distribuirse usando cualquier sitio cuyas condiciones de servicio obligasen a la concesión de usos comerciales de lo que se publicase con ellos. Para eso, hay que leerse las condiciones de aceptación del servicio. Pero el *Internet Archive*, *Scribd*

o *YouTube* exigen ese uso comercial. Son sólo ejemplos, aunque la lista sería seguro casi infinita.

### SIN OBRA DERIVADA

La segunda condición no permite obras derivada de la obra original. En el fondo, este esfuerzo pretende ser individual, no colaborativo. El motivo es doble: tanto por la cosa de la que trata, como por la persona que lo desarrolla. Creo que hay una parte de liderazgo en todo proyecto de colaboración, que exige un tiempo, un saber hacer y una energía que no tengo.

El tiempo es el recurso más escaso que todos somos. Por tanto, prefiero dedicarme sólo a escribir, que es lo propio del autor de un libro. Además, en el peor de los casos, si este libro no tuviese ninguna utilidad para nadie, lo sabría antes.

Respecto al tema de que tratan estas páginas, creo muy sinceramente que es mejor escribir un libro distinto partiendo de cero que intentar mejorar otro. Porque así habrá más recursos de donde elegir. Y podrán ser mejores, no sólo una mera repetición de lo que ya hay.

### SÓLO DESCARGA

La tercera condición es una razón totalmente práctica. Sólo se permite la descarga gratuita, para compartir el libro está la dirección `http://www.aprender-context.tk`. Por favor, si quieres poner el libro en internet, pon simplemente el enlace anterior.

La razón para esa decisión es la siguiente. Estas páginas no son una novela. En la parte del código, podría llegar a haber errores. Puede haber modos nuevos o distintos de hacer cosas. Lo que es seguro es que este libro cambiará. De momento, porque estoy todavía redactándolo. Es sólo un borrador. Después, como obra acabada, quiero que haya

mejoras según sugiera la gente que lo lea. Cada nueva versión hace obsoleta a la anterior.

La última versión es importante, tanto en la versión digital, como su impresión en papel. Personalmente, creo que este libro se verá bien en pantalla —de al menos diez pulgadas—. Imprimirlo es decisión de quien lo vaya a leer. Como contendrá bastantes páginas, siempre es mejor imprimir la última versión disponible.

### UN PACTO DE NOBLEZA

Este libro pretende ser un proyecto útil. Mi motivación es devolver a quienes hacen posible ConTEXt un poco de lo mucho que he recibido de su saber hacer. Es la mejor manera que tengo de mostrar mi agradecimiento: poder hacer que otras personas usen ConTEXt.

Quien lea este libro puede encontrarse con las tres situaciones que describo a continuación. Me permito sugerir la que entiendo que puede ser la mejor respuesta a cada una de esas posibilidades.

- El contenido de este libro es francamente malo. Como autor, siento que este libro no cumpla su objetivo. Como persona, no tengo interés en hacer perder tiempo a nadie —sólo hay pérdida en eso—. Por si fuese útil, agradecería un comentario explicando por qué el libro es malo[38].

- Estas páginas tienen puntos mejorables. En ese caso, es muy importante explicar en qué consisten los fallos y por dónde debería ir una posible mejora.

- *Aprender ConTEXt. Una introducción a la tipografía digital* es realmente útil. En este caso, agradecería un comentario contándolo. Y quizá lo más importante sea recomendarlo a personas que pensemos que les puede ser útil.

### COMENTARIOS

Si tienes algo que decir sobre el libro, te agradecería una incidencia en la dirección `https://github.com/ousia/aprender-context/issues/new`. Se requiere la inscripción previa en *GitHub*. Todos los comentarios son públicos. Por desgracia, el sistema está desarrollado de tal manera que no permite comentarios privados.

La crítica es libre. La única condición es que sea respetuosa con las personas. No habrá moderación previa de los comentarios. Sin embargo, si un comentario no es respetuoso, podrá desaparecer.

## AGRADECIMIENTOS

Como es ya sabido, estamos siempre a hombros de gigantes. Es una imagen clásica, que entre otros Newton cuenta en una de sus cartas [39]. Si la misma humanidad no deja de ser una gran conversación, sería necio por mi parte no reconocer la gran deuda de gratitud contraída con muchas personas. No es siquiera sensato que nombre a todas, pero es de justicia que mencione a las más importantes.

En cuanto a ConTEXt mismo, tengo que destacar la labor de Hans Hagen. No sólo desarrolla el programa de modo infatigable y constante. También atiende a las propuestas razonables de mejora y las incorpora en muy poco tiempo. En mi caso, sé que ha incluido en ConTEXt más mejoras de las que mi mala memoria puede recordar. Además de eso, está disponible en la lista del programa para las preguntas que hacemos los usuarios. Este libro es un modo tangible de mostrar mi agradecimiento por todo lo que es ConTEXt.

Wolfgang Schuster merece también un agradecimiento muy especial por parte de la comunidad —pequeña o selecta, según se prefiera— de usuarios de ConTEXt. Es admirable su capacidad de ayuda, en rapidez y calidad de las soluciones y explicaciones [40]. No tengo ningún reparo en reconocer que sin su ayuda no usaría ConTEXt en la actualidad. Creo que todos los usuarios de ConTEXt le debemos mucho a sus respuestas y a sus contribuciones de mejora al código.

Sin embargo, mi comienzo en la tipografía digital se lo debo a Αντώνης Τσολομύτης. Hace ya más de tres lustros me introdujo con paciencia en el mundo de TEX. No recuerdo cada uno de los detalles de algo tan relativamente lejano en mi biografía. Pero no olvido mi frustración y mi total incomprensión de qué podía ser aquello que era tan distinto de *Microsoft Word*, del que tan cansado estaba [41]. Gracias a

su saber experto, conseguí poder escribir textos en español, con pasajes en otros idiomas —especialmente, griego clásico—. Sin su ayuda, hoy no sabría qué podría ser eso de tipografía digital. Hubiese tirado todo por la borda y seguiría usando cualquier procesador de textos.

Durante ya más de quince años de uso de TEX, ha sido una ayuda fundamental la lista de correo de hispanohablante ES–TEX. Reconozco muy agradecido la deuda contraída con personas sabias, por sus consejos. Con esas indicaciones he aprendido mucho de tipografía, de ese arte que es la composición de imprenta.

En un plano personal, tengo que agradecer mucho a las personas que tengo la suerte de llamar amigos. Por supuesto, habría muchas más personas a quienes le debo mucho, pero quiero centrarme en aquellas que han sido especialmente relevantes en la gestación de este libro.

A Carlos le tengo que agradecer, además de su amistad auténtica, dos cosas. El primer asunto fue hacer que me diese cuenta de podía escribir. Contra lo que pensaba, debía escribir porque no se me daba del todo mal. Sus palabras fueron mucho más generosas, pero no me acabo de sentir cómodo con ellas. Entre otras cosas, lo de ponerme a escribir incluía *Aprender ConTEXt*. La segunda cuestión ha sido haber leído desinteresadamente los primeros borradores de este libro. Su generosidad está fuera de toda medida, porque no tiene interés alguno en ConTEXt. Gracias a él puedo saber que las partes que ha leído no son especialmente indigestas.

A Pablo le debo el valor de una amistad constante, mantenida a lo largo ya de casi media vida. He tratado de ayudarle en su trabajo informático con textos, pero le he mantenido alejado de ConTEXt por una buena razón. Sólo tiene sentido usar ConTEXt si tienes interés por el resultado final. Cuando no puedes controlar la publicación, la composición tipográfica la tendrá que hacer quien produzca ese resultado. Su ayuda en infinidad de detalles ha sido decisiva en multitud

de ocasiones. Espero que estas páginas puedan llegar a servirle algún día.

A Rosa le debo su confianza de amiga y una esperanza más allá de toda posible fe. A su gran capacidad intelectual y de trabajo se le añaden un auténtico interés por aprender. Espero que este libro le pueda llegar a servir para que pueda aprender todavía más de lo mucho que sin duda ya sabe. Y que también aporte algo a lo mucho que me ha dado ella.

Con Óscar tengo una deuda de amistad que es ya antigua. Su confianza en mi capacidad de escritura es mayor de la que yo haya tenido nunca. Sus sabias palabras me han ayudado en más de una ocasión. Me ha permitido experimentar que con nuestros amigos podemos cosas que por nosotros mismos no seríamos capaces.

A Edu le tengo que agradecer, además del honor de poder llamarme amigo suyo, sus acertados consejos. Entre otros ámbitos, sobre diseño gráfico y tipografía digital. A él le dedico estas páginas, con la esperanza de que las llegue a usar algún día. O quizá mejor —la ociosidad brilla por su ausencia en una vida lograda—, con la esperanza de que encuentre el tiempo para siquiera poder leer este libro con cierta calma.

# ANTES DE EMPEZAR

# 1 INSTALAR CONTEXT



Para poder usar ConTEXt, como para cualquier herramienta informática, tenemos que instalarlo primero. Por supuesto, podemos probarlo en su versión ejecutable en internet [42]. Así nos serviría sólo para un uso ocasional y bastante precario. Si queremos aprender, hemos de instalar ConTEXt en un ordenador que usemos habitualmente.

Para llegar a tener ConTEXt instalado en nuestro ordenador, tenemos dos posibilidades, que no se excluyen mutuamente:

- Tomarlo directamente de una compilación de TEX, como pueden ser *TEX Live*, *MacTEX*, *proTEX* o *MikTEX* [43]. Es útil si queremos usar alguna otra versión de TEX, además de ConTEXt. Si esto suena nuevo, es bastante probable que ésta no sea la opción.

- Instalar directamente la versión denominada *ConTEXt Suite*. Es la manera de tener la versión más actualizada. Es de obligado uso, si realmente buscamos sacarle el mayor partido o meternos en serio en ConTEXt.

En este capítulo, explicaré únicamente la segunda opción, la que trata de la instalación de *ConTEXt Suite*. Existen varios motivos. El motivo más importante es que la versión más actualizada de ConTEXt la encontramos en *ConTEXt Suite*. Por supuesto, nada nos impide instalar una distribución de TEX. Pero si vamos a usar únicamente Con-

TEXt, nos sobrará la grandísima mayoría de datos que contiene. Y si ya usamos TEX u otro de sus derivados, sólo tendremos que instalar *ConTEXt Suite*.

## A VERSIONES Y DESARROLLO DE CONTEXT

ConTEXt es un programa que está en constante evolución. Y tiene dos ramas, una estable y otra de desarrollo. La rama estable tiene una versión nueva aproximadamente cada doce meses. La rama de desarrollo tiene versiones casi a diario. La teoría dice que las versiones estables no tienen fallos —al menos conocidos— y que las versiones de desarrollo pueden hacer que nada funcione.

La teoría no difiere de la práctica, según la teoría. Sin embargo, siempre son distintas en la práctica. ConTEXt es un proyecto gigante que desarrolla fundamentalmente una persona: Hans Hagen[44]. La versión de desarrollo es la que está en mantenimiento. La versión de desarrollo es bastante estable. Claro está, excepto cuando se avisa de que se van a hacer experimentos y de que puede haber fallos. También puede haber fallos puntuales, que normalmente se corrigen cuando se advierte de que existen.

La versión estable en realidad es la versión de desarrollo que, para su inclusión en *TEX Live*[45], se revisa especialmente que no tenga fallos. Pero si existe alguno menor, el fallo no se corrige en versión estable. Si todavía existe el fallo en la versión de desarrollo, se corrige en ésta.

## B ¿POR QUÉ APRENDER CON UNA VERSIÓN EN DESARROLLO?

El desarrollo de ConTEXt muestra que las versiones en desarrollo son versiones estables en cierta manera. Como usuario, me arriesgo a afirmar que la versión en desarrollo es relativamente estable. Es relativamente estable para un uso habitual sin sorpresas.

Yo uso exclusivamente la versión de desarrollo que procuro actualizar siempre que tengo opción —suele ser una vez al día—. Lo hago por dos motivos fundamentales: lo positivo y lo negativo. La programación informática — en este caso, el desarrollo de ConTEXt por Hans Hagen— es un proceso, no es un estado o un hecho. Por eso, en la medida de lo posible, creo que es mejor no usar versiones obsoletas. La parte positiva consiste en las funcionalidades que sólo las nuevas versiones tienen. Y la parte negativa se cifra en los posibles fallos, que pueden estar corregidos en versiones más nuevas. Del mismo modo que existen fallos [46] que habrá que detectar y corregir en la última versión de desarrollo.

Como ConTEXt tiene un proceso de desarrollo bastante ágil, las funcionalidades que se van implementando en las versiones de desarrollo son relevantes. Por eso merece la pena usar la versión de desarrollo. No son sólo versiones que hay que expurgar de fallos. Son versiones que tienen lo último. Es importante advertir que el ciclo de versiones estables es anual.

La ventaja no sólo se nota en nuevas funcionalidades o en corrección de fallos. También se nota en rapidez de proceso. Con un ejemplo de este mismo libro, en el momento en que no ocupa más de 92KB y el documento PDF contiene 88 páginas. Generarlo completamente con la última versión de desarrollo de ConTEXt lleva 20 segundos. La versión estable de *TEX Live 2013* necesita casi 35 segundos. Es más rápido, aunque no sea una medición exacta [47]. También tengo que decir que hay un pequeño cambio en el documento por una nueva funcionalidad en la carga de tipografías, por la que la versión de desarrollo gana tres segundos respecto a la versión estable [48].

De todas formas, quien no quiera la versión de desarrollo no tiene especiales problemas. *ConTEXt Suite* permite la instalación en un mismo ordenador de diferentes versiones —estable y en desarrollo—.

Porque, como veremos, son independientes. Por si alguien duda, en este momento tengo en mi portátil, además de la última versión de *TEX Live* completo y la última versión de desarrollo de *ConTEXt Suite*, casi otras sesenta y cinco versiones de distintas fechas de la versión de desarrollo [49].

## C UNA DISTRIBUCIÓN PORTÁTIL

ConTEXt no es propiamente un archivo ejecutable que acceda a otros archivos llamados bibliotecas [50]. En la práctica, el núcleo es un conjunto de más de tres mil setecientos archivos. Sólo el núcleo no funcionaría, porque serían sólo instrucciones. Las instrucciones necesitan de un programa de ejecución tipográfica —en inglés, *typographic engine*—, como LuaTEX [51]. Todo junto, acompañado de algunas tipografías digitales, es lo que se llama una distribución.

*ConTEXt Suite* es una distribución que contiene más de ocho mil quinientos archivos. Aunque pueda impresionar, en realidad la versión completa se descarga en menos de setenta segundos. Para esa descarga, sólo hace falta un ordenador correctamente configurado y una conexión a internet decente [52]. En las actualizaciones, no hay movimiento de datos en todos los archivos en los que no haya cambiado nada.

Una distribución portátil es aquella que no necesita escribir datos en el sistema operativo para funcionar. Como consecuencia práctica, se puede borrar el directorio —la carpeta en un sistema visual— que lo contiene para desinstalar completamente *ConTEXt Suite*. Otra ventaja es que se puede copiar de un ordenador a otro, copiando el directorio en que lo hemos instalado [53].

Propiamente, *ConTEXt Suite* no se instala, porque no escribe datos en el sistema. Esa es la característica definitoria de una distribución portátil. Ésa es la razón —como te contaba antes— de que tenga más

de sesenta versiones del mismo programa en mi ordenador sin que haya problemas. Sólo pueden coexistir porque realmente no están instaladas.

## D INSTALACIÓN DE *CONTEXT SUITE*

ConTEXt es un programa que funciona directamente por órdenes, no por ventanas. Tenemos que abrir la ventana de órdenes y teclear. También para instalarlo, porque no existe un instalador gráfico. Aunque eso nos pueda impresionar un poco, la parte buena está en que para instalar *ConTEXt Suite* sólo hay que descargarlo. Una vez descargado, realmente no hay que instalarlo. Sólo hay que descomprimirlo y ya está listo para usar. Te recuerdo otra ventaja: con borrar el directorio que contiene *ConTEXt Suite* no quedarán rastros del programa [54].

La installación resultará obvia para quienes tengan cierta experiencia de uso de ordenadores. Si no la tienes, tendrás que aprender a hacer algo nuevo. No pasa nada. Aunque todavía no sepas, te puedo asegurar que no es difícil. Sencillamente hay que seguir las instrucciones. Serán lo más sencillas que me sea posible. Todo está en darse cuenta que con teclado también se puede manejar un ordenador. Sólo es hacer las cosas de otra manera. Y es bueno acostumbrarse a esa nueva forma. Doy por supuesto que quien lea sabe descomprimir un archivo [55].

He de advertir que el sistema operativo que uso habitualmente es *Linux*. Tengo acceso a ordenadores con *Windows*, pero no a ordenadores con *MacOS X*. Si me permites la ironía, no debo tener la clase para poder acercarme a *MacOS X*.

### INSTRUCCIONES COMUNES

El método de instalación que explico es para un usuario individual [56]. La diferencia con la instalación para todo el sistema es que tengamos permisos de lectura y escritura en los directorios en los que

ConTEXt tiene que escribir archivos temporales y de configuración. Si estás empezando, mejor hacer la instalación individual.

Existen varios programas que ConTEXt usa para ejecutar todas sus funcionalidades[57]. El único programa realmente necesario para instalar *ConTEXt Suite* es `rsync`, que está incluido con la descarga para *Windows*. Aunque en otros sistemas operativos es necesario tenerlo instalado previamente.

*ConTEXt Suite* necesita descargar un poco más de 250MB de archivos. Es necesario disponer de un poco más de espacio libre. Porque los archivos de configuración pueden llegar a ocupar casi 50MB más. Como recomendación general, no aconsejaría instalar *ConTEXt Suite* con menos de 500MB disponibles.

El último requisito común es la necesidad crear el directorio —la carpeta de un entorno de escritorio— en el que vamos a instalar *ConTEXt Suite*. En *Windows* se crea sólo al descomprimir el archivo que hay que descargar. Lo veremos con sus instrucciones propias.

Por último, es útil señalar que en cualquier sistema operativo puede descargarse la versión para otro sistema operativo distinto. De hecho, las versiones para *Windows*, yo las descargo en *Linux*. Para evitar incomprensiones, no es que cualquier versión de *ConTEXt Suite* de cualquier sistema operativo sirva para cualquier otro. Simplemente se puede descargar la versión para otros sistemas operativos —como explico en *1.G. Descarga para otro sistema operativo*—. Por supuesto, sólo podremos usar la versión propia del sistema operativo que tenga el ordenador en el que trabajemos.

### *WINDOWS*

En *Windows* es necesario descargar un archivo comprimido[58]. Una vez descargado, hay que descomprimirlo. Mi consejo personal es que

tengas ConTEXt en el escritorio. Para eso, lo que tienes que hacer es descomprimir el archivo del párrafo anterior allí. Y dejarlo así, sin moverlo. Puedes ponerlo donde quieras, pero tendrás que saber hacerlo.

Una vez hecho, el directorio `context` en el escritorio, tienes que abrir la ventana de línea de órdenes. Puedes pulsar el botón de Inicio, luego ir a Programas, Accesorios, Herramientas del sistema y seleccionar Símbolo del sistema. O con un atajo de teclado, pulsa la tecla `r` mientras mantienes pulsada la tecla de *Windows*. Surgirá una ventana con la línea para escribir, en la que escribes `cmd` y pulsas el botón Aceptar.

Una vez abierta la ventana de órdenes, tienes que teclear[59] —o puedes copiar y pegar—:



```
cd %USERPROFILE%\Desktop\context\ && first-setup.bat
--modules=all && tex\setuptex.bat && mtxrunjit
--generate
```

Los pasos sucesivos hacen lo siguiente:

1. Los signos `&&` unen órdenes. Sólo se ejecuta la orden posterior, si la anterior no da un mensaje de error.

2. Va al directorio `context` que tienes en el escritorio.

3. Ejecuta el programa de descarga de ConTEXt.

4. Carga las rutas de ConTEXt.

5. Genera la base de datos de archivos que ConTEXt necesita para trabajar.

Después de eso, sólo tienes que probar que la instalación funciona correctamente.

MACOS X

*MacOS X* es un sistema derivado de *Unix*, al igual que *Linux*. Su modo de instalación es el mismo, con una pequeña diferencia. Lo explico en la siguiente sección.

LINUX

Antes de empezar, una advertencia exclusiva para *Linux*. En el caso de este sistema operativo, es fácil que con el gestor de paquetes de tu distribución te venga algún paquete que sea ConTEXt. Con sólo teclearlo, lo tendrías instalado. Pero sería la versión de *TEX Live*. ¿Puedes usarla? En realidad, no deberías, porque puede ser antigua o muy antigua. Como te he explicado en *1.B. ¿Por qué aprender con una versión en desarrollo?*, ConTEXt tiene versiones muy frecuentemente con correcciones y nuevas funcionalidades. Si decides usar la versión de *TEX Live* —o no actualizas *ConTEXt Suite*—, es posible que algún punto de estas páginas no funcione como esperes.

La gran noticia es que puedes instalar *ConTEXt Suite* con *TEX Live* en el sistema. Si no tienes problemas de espacio en el disco duro, puedes tener todas las versiones anteriores de desarrollo[60]. Como ya te he contado, no interfieren entre sí. Antes de poder proceder a la instalación en *Linux*, debes haber instalado —según el método habitual en tu distribución— `rsync`[61].

La manera más fácil de instalación es usar la siguiente línea:



```
mkdir ~/context && cd ~/context/ && rsync -ptv
rsync://contextgarden.net/minimals/setup/first-setup.sh
. && ./first-setup.sh --modules=all && echo \ >>
tex/setuptex && echo \ >> tex/setuptex && echo
"export OSFONTDIR=\"/usr/share/fonts//\"" >>
```

```
tex/setuptex && source tex/setuptex && mtxrunjit
--generate && cd
```

La versión de las órdenes para *MacOS X* es ligeramente distinta [62]. Puedes copiarla y pegarla directamente en una ventana de órdenes. Pero antes, te pediría que intentases entender lo que vas a hacer. Te explico cada uno de los pasos:

1. Lo que une las diferentes órdenes es `&&`. Ejecuta la orden posterior sólo cuando la anterior se ha ejecutado completa y sin errores. Es útil para hacer todas las operaciones copiando y pegando una única vez.

2. La orden `mkdir ~/context` crea un directorio —una carpeta en un sistema de escritorio— en el que instalaremos ConTEXt.

   Por supuesto, lo puedes instalar donde quieras. Te aconsejo que lo hagas ahí. Si cambias, recuerda que tienes que adaptar todo lo demás.

3. Vamos al directorio de ConTEXt.

4. `rsync` copia correctamente todos los archivos necesarios de *ConTEXt Suite*. También se usa para actualizar.

5. Ejecutamos el programa —`first-setup.sh`—, especificándole que descargue todos los módulos.

6. La triple orden posterior —empieza con el primer `echo`— añade una línea al archivo de configuración para que también lea los tipos digitales del sistema operativo. En *Windows* o *MacOS X*, está ya especificado. Pero no en *Linux* [63].

7. Tienes que cargar las rutas de ConTEXt para poder hacer el paso siguiente. Lo hace la siguiente orden:

```
source ~/context/tex/setuptex
```

`~/context/` será el directorio donde tengas instalado ConTEXt. Si es otra ruta, tendrás que especificar otra. Insisto en que no la cambies, a no ser que tengas un motivo muy claro. `tex/setuptex` es la parte inamovible, porque la estructura de *ConTEXt Suite* es así.

8. Para que ConTEXt pueda empezar a trabajar por primera vez, tiene que generar la base de datos de archivos del propio programa. Por eso tienes que ejecutar `mtxrunjit --generate`.

9. Por último, es bueno volver al directorio de usuario tecleando `cd`.

De este modo, estaría copiado *ConTEXt Suite*. Sólo queda probar que funciona.

## E COMPROBACIÓN FINAL DE FUNCIONAMIENTO

En este momento, una vez descargada e instalada la distribución de *ConTEXt Suite*, es necesario comprobar que funciona. La manera más fácil es mostrar las versiones de los programas en la ventana de órdenes.

En *Windows* sería:



```
%USERPROFILE%\Desktop\context\tex\setuptex.bat &&
contextjit --version && luatex --version
```

En los diferentes estilos de *Unix* —*MacOS X* y *Linux*— habría que teclear:



```
source ~/context/tex/setuptex && contextjit --version
&& luatex --version
```

Lo que estarás haciendo con las diferentes órdenes es:

1. Cargar las rutas de *ConTEXt Suite*. Es necesario este paso porque es portátil. Así tampoco tendrías interferencias con *TEX Live*, si lo tuvieses instalado.

2. Pedir la versión de ConTEXt instalada.

3. Mostrar información sobre LuaTEX[64].

Si todo está correcto, verás dos mensajes sobre las versiones de ConTEXt y LuaTEX. En ese caso, ya estás listo para empezar con los *Primeros pasos*. Aunque no es mala idea que sepas cómo tienes que actualizar el programa.

## F ACTUALIZACIÓN

Como te he dicho, ConTEXt se actualiza con una frecuencia relativamente alta. Lo normal es una vez por semana, pero no es raro que haya versiones nuevas con más frecuencia. Para saber cuál es la última versión, puedes visitar `http://www.pragma-ade.com/download-2.htm`[65]. El primer archivo es la versión de desarrollo y tiene hora y fecha.

La actualización para *Windows* es:



```
robocopy
%USERPROFILE%\Desktop\context\tex\texmf-cache\*
%USERPROFILE%\Desktop\ /move /e && cd
%USERPROFILE%\Desktop\context\ && first-setup.bat
--modules=all &&  tex\setuptex.bat && mtxrunjit
--generate && cd
```

Para *MacOS X*, las órdenes serían:



```
mv -i ~/context/tex/texmf-cache/* ~/Desktop && cd
~/context/ && ./first-setup.sh --modules=all &&
source tex/setuptex  && mtxrunjit --generate && cd
```

Para *Linux*, hay una pequeña diferencia:



```
mv -i ~/context/tex/texmf-cache/* ~/Desktop && cd
~/context/ && ./first-setup.sh --modules=all && echo
\ >> tex/setuptex && echo \ >> tex/setuptex && echo
"export OSFONTDIR=\"/usr/share/fonts//\"" >>
tex/setuptex && source tex/setuptex  && mtxrunjit
--generate && cd
```

Explico lo que haces en todos los casos:

1. Mueves el directorio `luatex-cache` al escritorio. Desde allí, lo puedes borrar a mano[66].

2. Vas al directorio de ConTEXt.

3. Ejecutas el programa de descarga, que copiará sólo los archivos necesarios.

4. Sólo en *Linux*, hay que añadir el directorio de los tipos de letra del sistema operativo.

5. Cargas las rutas de ConTEXt.

6. Ejecutas el programa para que puedan crearse los archivos de configuración —borrados en el primer paso—.

## G Descarga para otro sistema operativo

El programa de descarga de *ConTEXt Suite* detecta automáticamente cuál es la versión que tiene que descargar en el ordenador en que lo

hace. Sin duda, eso es utilísimo. Aunque también es muy útil poder obligar a descargar una versión para un sistema operativo concreto. Porque así puedes descargar, por ejemplo, las versiones de *Windows* en *Linux*. Es algo que hago de vez en cuando y me es mucho más cómodo que hacerlo directamente en *Windows*.

Por supuesto, asumo que si decides probarlo es porque sabes lo que haces y lo necesitas. Me explico. Es importante que descargues la versión de *ConTEXt Suite* que necesitas para aprender. Si descargas otra que no vas a usar —o que no te sirva para nada—, seguro que de ConTEXt no aprenderás nada.

La opción para obligar a la descarga de la versión de un sistema operativo concreto es `--platform`[67], como en:

```
first-setup.sh --modules=all --platform=win64
```

Con el ejemplo anterior, especificarías que la versión descarga sea *Windows* con arquitectura de 64bit.

# 2 ENFRENTARSE AL CÓDIGO



Para quien no tiene costumbre de ver el código de texto formateado, la visión le puede llegar a impresionar. De ese primer contacto puede nacer incluso miedo, que se traduce en mantener una distancia respetuosa. Te puede parecer que es imposible entender nada. Puedes alimentar ese miedo al código o puedes aprender a leerlo. Es más sencillo de lo que parece. Sólo no huyas de lo que no entiendes. El objetivo de este capítulo es darte las claves que te permitan descrifrar el código de ConTEXt. Con la práctica, verás que es muy fácil entenderlo.

## A PRIMER FRAGMENTO DE CÓDIGO

Un documento muy básico y completo en ConTEXt podría ser el siguiente:

```
\mainlanguage[es]
\setuppapersize[A5]
\starttext

\chapter{Introducción}

\ConTeXt\ es {\en\em a document generation system} en
palabras de {\sc Hans Hagen}.
\stoptext
```

Aunque nos pueda parecer ininteligible, es muchísimo más comprensible que cualquier página de nuestro navegador de internet o

procesador de textos favorito. Aunque pueda parecer lo contrario, el código de un texto con formato es más bastante claro que el resultado del texto ya formateado. Sólo tenemos que aprender los pocos detalles que nos impiden entenderlo.

### B DESCRIFRAR EL CÓDIGO

No es momento ahora de explicar el código anterior. Ésa es la tarea que tienen por delante el resto de capítulos. El fragmento anterior sólo es un ejemplo para mostrar lo siguiente: si tenemos interés en aprender ConTEXt, lo primero que tenemos que hacer es perderle miedo al código. No sólo porque lo tendremos que descifrar, sino porque lo tenemos que escribir cada cual. Y lo escribiremos, si este libro no está muy mal escrito.

Todo el código de ConTEXt contiene el texto mezclado con partes que especifican cómo tiene procesar el programa ese texto o fragmentos de éste. Esas partes se suelen denominar marcas, etiquetado o sencillamente órdenes. Para que podamos reconocerlas, paso a describirlas.

### C MAYÚSCULAS Y MINÚSCULAS EN EL CÓDIGO

Es bastante posible que a nosotros nos parezca igual, pero no lo es. Todo el código que te voy a explicar tiene la siguiente característica: distingue entre mayúsculas y minúsculas. Pasa siempre con los ordenadores. Por tanto, usar mayúsculas o minúsculas al escribir código da resultados distintos.

Un ejemplo de algo que vamos a ver en la siguiente sección: `\orden` y `\Orden` son cosas distintas. Cuidado porque, en el mejor de los casos, las cosas funcionarán de manera distinta a como deberían. Y la gran mayoría de veces no funcionarán.

La explicación es muy sencilla. Los ordenadores organizan información, no entienden sentido —como mucho, pueden simular que lo hacen—. En español, *u*, *U*, *ú*, *ù*, *ü* o *Û* son la misma letra. Para un ordenador, únicamente son caracteres distintos. En la escritura real, mayúsculas y minúsculas son modos distintos de escribir la misma letra. Para una calculadora que tiene básicamente unos y ceros[68], lo de la letra es secundario. Por eso, escribir en mayúsculas o en minúsculas es escribir cosas distintas.

## D ÓRDENES

La forma más básica del etiquetado en cualquier sistema de marcado de textos es la orden[69]. Una orden especifica un mandato al programa. De forma muy básica, cuando usamos una orden, le decimos al programa: «haz esto». Toda orden tiene la forma:

```
\orden
```

Una orden es toda secuencia de letras que va desde la barra invertida —\— hasta un espacio, un signo de puntuación o un salto de línea. Por poner un par de ejemplos reales —sólo nos tenemos que fijar en la forma, no en qué pueden hacer—, son órdenes[70]:

```
\completecontent \ConTeXt. \columns
```

## E OBJETO

Las órdenes citadas hasta ahora —como `\ConTeXt`— pueden denominarse simples. Especifican es que el programa realice una tarea determinada. La orden de la frase anterior es la que permite escribir ConTEXt como un logotipo —le decimos al programa: «inserta el logotipo de ConTEXt»—.

Probablemente estas órdenes simples sean las menos frecuentes en ConTEXt. Las órdenes simples no tienen objeto, porque no lo necesitan cuando se usan. Sin embargo, es más habitual pretender que un texto se presente de una determinada manera, como poner en cursiva un texto, o especificar el título de un capítulo. Eso es el contenido de la orden[71]. El contenido u objeto de una orden se expresa entre llaves —como en `{objeto}`—.

Todo contenido u objeto pertenecen a una orden. Algunas órdenes pueden no tener objeto. Otras necesitan tener objeto. Pero un objeto necesita siempre una orden. Si no hay orden, el objeto o contenido lo serían de nada. El objeto va siempre con su orden correspondiente, como en estos ejemplos:

```
\orden{objeto}
```

```
\chapter{El título del capítulo que quería}
```

No es obligatorio que todas las órdenes tengan que tener objeto. Ni tampoco tienen que tener sólo uno. Por ejemplo, supongamos que queremos escribir un texto que tenga un vínculo a una página de internet en el documento PDF final. Si bien la orden no está creada en ConTEXt —habría que definirla antes—, su uso sería algo así como:

```
\href{http://www.aprender-context.tk}{Aprender
\ConTeXt}
```

### F OBJETO POR AGRUPACIÓN

Hay órdenes que propiamente no tienen objeto, porque realizan un cambio en el texto desde el punto en están escritas. Pero se les puede asignar un objeto al limitar el ámbito de aplicación con un par de llaves —`{}`—. Con las llaves agrupamos el objeto de la orden, limitamos la porción de texto en la que la orden va a tener efecto.

Fundamentalmente se usa con cambio de idiomas o tipo de letra. Con idiomas y tipos de letra, es esencial limitar su aplicación a un único fragmento. Tomemos dos ejemplos básicos:

```
{\em objeto}
```

```
{\de Übergang}
```

En sentido estricto, se trata de órdenes simples, sin objeto. Pero hay situaciones en que es necesario que se limiten a un objeto. Tomando los ejemplos anteriores, si especificas que el texto esté enfatizado desde la orden o que el idioma sea alemán, necesitarás limitarla. El modo de limitarla es encerrar la orden y su objeto entre llaves. Exactamente como en los dos ejemplos anteriores de código.

## G OPCIONES

En algunas órdenes simples, a veces es posible configurar cómo queremos que se realice el mandato de la orden. Se trata de órdenes de configuración. Por supuesto, una orden puede configurarse si tiene varias opciones de las que poder elegir otra distinta de la predeterminada. La configuración siempre va entre corchetes, siendo su forma:

```
\orden[opción, opción=valor]
```

Como vemos, las opciones pueden ser de dos tipos: simples o con asignación de valores. Por supuesto, cada orden admite sus opciones propias de configuración, aunque pueda haber órdenes que compartan opciones. Poniendo ejemplos reales:

```
\setuppapersize[A5]
\setupinterlinespace[line=3.2ex]
\setupindenting[medium, always]
\definefontfeature[default][default]
    [script=latn]
```

Dependiendo de cada orden, pueden tener una o varias opciones. También algunas órdenes puede que especifique su configuración en más de un par de corchetes, como se muestra en el último ejemplo anterior.

### H ÓRDENES DENTRO DE ÓRDENES

Como todo programa informático, ConTEXt lee y ejecuta órdenes. Y para que ese proceso se realice, es necesario que estén escritas de tal manera que no sean ambiguas. No es únicamente que estén escritas sin errores ortográficos, como `\capter` en vez de `\chapter`. Es que las órdenes con objeto sepan claramente a qué objeto refieren.

Se pueden combinar órdenes dentro de órdenes, como si de muñecas rusas se tratase. Pero no podemos cerrar el objeto de una orden —las llaves, `{}`— que contiene a su vez el objeto de otra orden, sin cerrar antes el objeto que está contenido en la orden. Para que esto no parezca un trabalenguas, pongo un ejemplo:

```
\footnote{Esto es una {\em nota al pie.}. Pero esto
no.}
```

Lógicamente, ConTEXt entiende que la primera orden tiene por objeto lo que se encuentra entre las llaves exteriores. Y las interiores entiende que son las que encierran la segunda orden. Es así de modo necesario, porque el ordenador lee lo que escribimos, no lo que pensamos[72]. El modo correcto de escribir las órdenes anteriores sería:

```
\footnote{Esto es una {\em nota al pie.}}. Pero esto
no.
```

Si quisiésemos la orden `\em` también después de la orden `\footnote`, tendríamos que escribir:

```
\footnote{Esto es una {\em nota al pie.}}. {\em Pero
esto no.}
```

# I SINOPSIS DEL ETIQUETADO

Este capítulo sólo pretende dar las claves para poder entender qué es qué en un texto formateado en ConTEXt. No qué significa cada cosa, sino qué clase de elemento es lo que no es el puro texto. Es una orden todo lo que empiece por el signo de barra invertida —`\`—. Y todo lo que no es texto escrito en el código es una orden. Ejemplos de órdenes son `\starttext` o `\stoptext`, que aparecen en todos los documentos de ConTEXt.

Algunas órdenes tienen objeto, delimitado entre llaves. Los objetos son parte del texto, pero que están limitados por una orden que les da un tratamiento especial. Son órdenes con objeto aquellas en que a la orden le sigue inmediatamente su objeto —`\chapter{Capítulo}`—, o las que están entre llaves —`{\em cursiva}`—.

La configuración es la parte de la orden inmediatamente posterior, que va entre corchetes. Puede tratase de una clave, una clave con valor asignado, y puede haber varios grupos de corchetes de configuración. Son órdenes válidas: `\mainlanguage[es]` o `\setuplanguage[es][patterns={es, agr}]`.

# 3 ¿POR QUÉ ES MEJOR TRABAJAR CON CÓDIGO?



Después del capítulo anterior, la tentación más habitual puede ser la siguiente: ¿seguro que es mejor trabajar teniendo que ver todo el código? Es cierto que para empezar con un nuevo programa, un programa gráfico que nos muestre visualmente el resultado es más fácil. O quizá parezca más fácil.

Sin embargo, uno de los puntos más débiles del enfoque visual, de que se nos muestre en la pantalla el resultado directamente, es que sólo se nos muestra lo que parece que hacemos, aunque en realidad, el código permanece oculto. Y es importante que el código sea accesible para ver qué estamos haciendo realmente. Porque así trabajaremos sin misterios, pudiendo llegar a saber —llegado el caso— si la limitación es nuestra o del programa.

## A NO ES SÓLO CONTEXT

En realidad, trabajar directamente con el código no es exclusivo de ConTEXt. No hablo de programación. Para trabajar con textos, lo habitual precisamente es que el texto esté en formato de código. De hecho, los procesadores de textos esconden ese código[73]. La diferencia es que los procesadores de texto no lo muestran. En el resto de casos, el texto con etiquetado es directamente accesible para quien lo escribe.

El ejemplo más claro es el código de las páginas de internet. XML es un modo de etiquetar textos para diferentes finalidades, entre ellas, para que sean páginas de internet. Por supuesto, podemos usar un

programa que nos haga la simulación, pero en el fondo, las páginas son los archivos de texto codificado en XML.

La diferencia de ConTEXt respecto a un navegador de internet es que éste último «interpreta» el código, mientras que ConTEXt lo «compila». Interpretar por parte de un navegador es leer el documento de origen y mostrarlo en pantalla según el formato especificado. La compilación que hace ConTEXt toma el documento origen y genera un documento PDF con las instrucciones que lee. Es ConTEXt quien escribe el documento PDF con las instrucciones que recibe del documento origen.

Pero el código fuente no sólo se muestra en ConTEXt o XML. También hay formatos de etiquetado llamados ligeros, que sirven para escribir con más facilidad documentos que luego estarán en otros formatos. Un ejemplo bastante conocido es el formato *MediaWiki*, que es el que se usa para la conocida *Wikipedia*. Para que los artículos de esta enciclopedia puedan tener formato, hay que escribirlos con unas determinadas convenciones[74].

Otro ejemplo de formato de etiquetado ligero es *Markdown* para pan doc[75] La idea de *Markdown* es poder generar de un único documento de origen la mayor cantidad posible de conversiones en otros formatos. Con pandoc pueden generarse de un único documento de origen, documentos de HTML, ePub, ConTEXt[76], LaTEX, *LibreOffice/OpenOffice Writer* y *Microsoft Word*, entre otros muchos.

En el fondo, trabajar con texto a un cierto nivel implica trabajar con el archivo informático que tiene el código que contiene al documento con formato. Porque de hecho el archivo origen es el documento de texto. Y al poder escribir directamente en el archivo origen, tenemos un acceso completo al documento. Sin acceso directo al código de origen, nuestro acceso estará limitado y nuestra capacidad de maniobra será restringida.

### B LA IMAGEN MECANOGRÁFICA

Cuando un programa no muestra el código y genera el resultado visual en pantalla —como un procesador de textos—, se sirve de una metáfora. Nos hace creer que el ordenador, al usar ese programa, es una máquina de escribir unida a una televisión. Esa metáfora contiene en sí sus propios límites.

La metáfora no se cifra en que el teclado sea un dispositivo de introducción de datos y el monitor sea un dispositivo que muestra datos. Ambas dispositivos son partes integrantes de un ordenador y tienen las funciones descritas. La metáfora estriba en la imagen de que un programa visual simula que estamos trabajando con una máquina de escribir. La única diferencia que tiene la imagen es que la hoja se sustituye por la pantalla. Aunque realmente en la pantalla se dibujan páginas.

### C TRABAJAR CON UN ORDENADOR

Hay una concepción bastante extendida de la informática que piensa que se trata de usar programas en vez de usar ordenadores. De este modo, saber de informática a nivel de usuario es tener más o menos automatizadas ciertas rutinas de cómo aplicar opciones de programas concretos en determinados casos. Aprender informática sería entrenar los automatismos con determinados programas.

Esa posición es muy precaria, porque no permite que entendamos qué hacemos con el ordenador. Las cosas funcionan o fallan como por arte de magia. O la máquina nos empieza mirando mal. Sorprendentemente, le acabamos cogiendo manía. Es bueno intentar entender la situación. Un ordenador es una cosa, no una persona. Tenemos que entender cómo funciona para poder sacarle el mayor partido posible. Lo mismo pasa con la batidora, nadie remueve el café de su desayuno con la misma batidora con la que hace purés.

Un ordenador es una máquina. Respecto a la máquina, los programas son herramientas que permiten realizar tareas en esa máquina. No es un mero mecanismo físico, por supuesto. Ni siquiera me refiero a la parte física de esa máquina, que la doy por supuesta. Un ordenador es una gran máquina lógica universal. Realiza toda clase de operaciones lógicas que no sean contradictorias.

Lo relevante de que un ordenador sea una máquina universal es que maneja programas. Por tanto, si sabemos usar el ordenador es más fácil que aprendamos a usar programas. Básicamente, porque sabremos cómo hace un ordenador las cosas. Creo que es muy distinto de esperar a ver qué pasa si hago algo. Y bastante más efectivo.

En el fondo, un ordenador es nuestro asistente digital. Respecto a ConTEXt, la máquina es nuestro cajista digital. ¿Qué quiere decir eso? Que hará la tarea, sólo si sabemos cómo mandarle. Sin atribuirle cualidades de personas, piensa en un siervo muy rápido y eficiente en tareas muy simples. No habla tu mismo idioma, sino una versión muy pobre de inglés. Si te entiende hará lo que le mandes más rápido y mejor que tú. El asistente no aprende nada. Pero puedes aprender tú a enlazar órdenes simples para hacer tareas complejas.

# 4 UN EDITOR DE CÓDIGO



## A AYUDA PARA ESCRIBIR

Como ya te he contado, ConTEXt lee archivos de código de los que genera documentos PDF. Esos archivos de código son los que escribimos las personas. Informáticamente, son archivos de sólo texto, también llamados archivos de texto puro. Todos los archivos que contienen código —que tienen órdenes informáticas escritas— son archivos de texto. Da lo mismo qué tipo de código sean y ConTEXt no es una excepción en este punto.

En el fondo, trabajar con archivos de texto puro es una ventaja. No necesitas un editor especial para abrirlos —como serían *Microsoft Word* o *LibreOffice Writer*—. Puedes usar el que más te guste, el que tengas más a mano, o con el que más cómodo estés. Con el tiempo, es lógico que en el mismo programa confluyan esas tres características. No tienes que usar siempre el mismo programa. Con el paso del tiempo, es probable que cambies de editor. Si usas sistemas operativos distintos, no tienes que usar el mismo editor de código en todos. La elección es importante, aunque única y exclusivamente por ti. Tú vas a ser el que va a teclear, ConTEXt no lo necesita para nada.

## B DOS FUNCIONES BÁSICAS

Escribir archivos de texto con órdenes no es sólo crear archivos nuevos —o abrir los existentes—, teclear y guardar los cambios. Si

recuerdas la imagen, los archivos de ConTEXt son sólo partituras. Es cierto que contienen notas, pero hay que tocarlas para escuchar la música. Eso le corresponde a ConTEXt, que interpreta los datos del archivo —órdenes y texto— para generar un documento PDF. Ese proceso es la compilación de órdenes y es automática. Siempre que le comuniques ConTEXt que la lleve a cabo y que no haya errores graves en las órdenes.

Las dos funciones fundamentales que vas a necesitar de un editor de texto para archivos ConTEXt son:

- Coloreado del código.
- Compilación automática.

## C COLOREADO DE CÓDIGO

El código de la nota al pie del capítulo anterior contiene el siguiente texto, que se vería:

```
\footnote{{\us “Computers are good at following
instructions, but not at reading your mind”}; {\sc D.
Knuth}, {\em The \TeX book}, Addison--Wesley, p.~9.}.
```

De ese modo lo mostraría un editor de textos normal. Todo el texto es absolutamente igual. Aunque órdenes y texto estén mezclados, el programa no los distingue. De esta manera, es más fácil cometer errores tecleando órdenes. También es más difícil detectar los errores que podamos llegar a cometer en las órdenes.

Un editor de texto especializado en código mostraría en color [77]:

```
\footnote{{\us “Computers are good at following
instructions, but not at reading your mind”}; {\sc D.
Knuth}, {\em The \TeX book}, Addison--Wesley, p.~9.}.
```

Así pueden distinguirse las órdenes —precedidas de la barra invertida y haciendo uso de llaves— del resto del texto. Las órdenes y sus llaves no aparecerán en el documento PDF. Pero son decisivos para que el texto esté correctamente en el documento final. Esa distinción por colores y negrita coloreada ayuda a prevenir errores. También ayuda a detectarlos, cuando se nos cuelen al teclear.

## D COMPILACIÓN AUTOMÁTICA

En la mayoría de sistemas visuales, se ordena al programa apretando algún tipo de botón. Por ejemplo, cualquier procesador de textos pone en cursiva el texto pinchando sobre el botón adecuado. De modo muy básico, ese botón lleva un texto y ésa es la orden real que se envía al sistema. Es muy cómodo pulsar botones. Si tienes que teclear, mi experiencia me dice que es más cómodo pulsar teclas. Por el simple hecho de que no te alejas del teclado en el que estás escribiendo.

ConTEXt existe para casi una decena y media de sistemas operativos. Aunque algunos ni siquiera sepas que existen, el modo de hacer que funcione en todos es mediante órdenes escritas. No hay botón, sólo una orden. Como veremos luego, esa orden es `contextjit`, a la que ha que añadir el nombre de archivo. Sin embargo, que sean órdenes escritas no impide que no se puedan llamar desde el editor donde tecleas con sólo una combinación de teclas o pulsando un botón.

La compilación automática ordena a ConTEXt que genere un documento PDF del archivo que estamos editando. Sólo tienes que pulsar una combinación de teclas[78]. El editor tiene que grabar los cambios en el archivo[79], ejecutar la orden, y de modo opcional mostrar el documento PDF resultado de la compilación.

La compilación automática es útil para no escribir la orden siempre que la necesites usar. En vez de teclear en una ventana de órdenes, desde el mismo editor mandas todo. ConTEXt se encarga de hacerlo.

## E ¿EDITORES PARA CONTEXT?

Existen muchos editores de texto que pueden servir para ConTEXt. Como depende en gran medida de cada persona y de sus conocimientos, cualquier recomendación debe quedarse en grado de sugerencia para probar. Si usas ConTEXt, el mejor editor es con el que tengas más facilidad y comodidad. No siempre tiene que ser el mismo. Con el tiempo y más experiencia, será lógico que cambies.

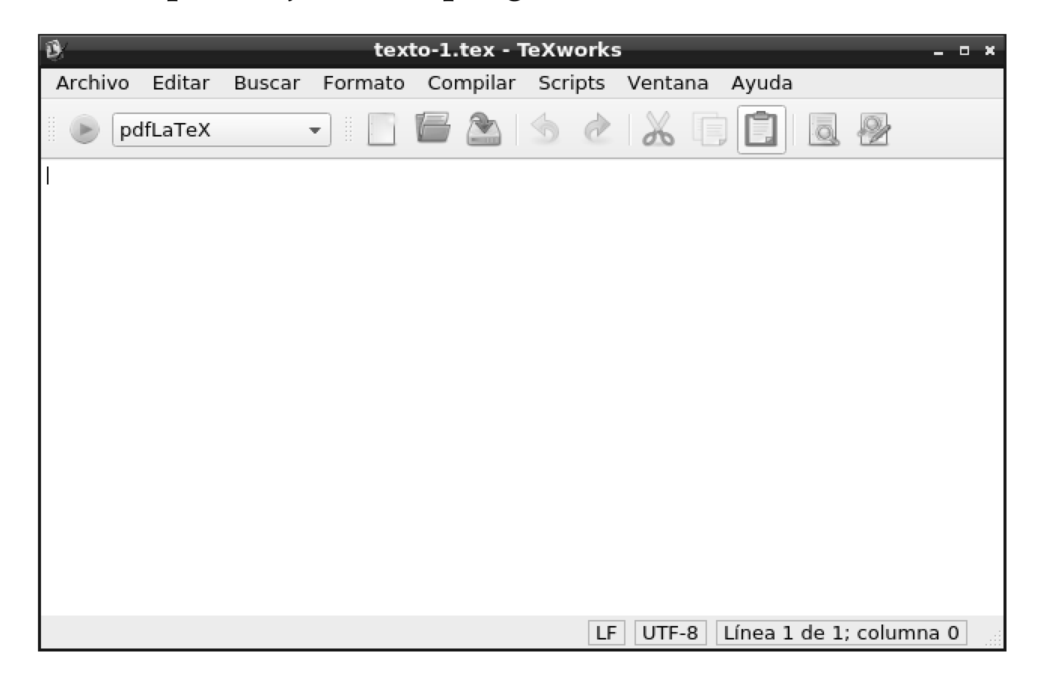
Sin que pretenda ser siquiera una indicación, yo uso *Geany*[80] en *Linux*. Cuando estoy con *Windows*, empleo *Notepad++*[81]. Con ambos consigo lo que necesito[82]. Sin embargo, no son especialmente recomendables para empezar, porque no están especialmente diseñados para ConTEXt. No cuesta mucho adaptarlos para poder usarlos así. Pero añades una dificultad más cuando comienzas a aprender. Existe un editor especializado para *MacOS X*, *TEXShop*. Quienes usan ese sistema operativo, lo recomiendan también para empezar con ConTEXt[83]. No dudo de las bondades de *TEXShop*, pero no puedo explicarlo. No tengo modo de usar un ordenador con *MacOS X*

De los editores de código haría dos grupos: editores de programación y editores para TEX. Los editores de programación suponen en muchos casos matar moscas a cañonazos. Tienen un gran potencial para otras tareas. Si no lo sabes aprovechar, pierdes el tiempo. En cuanto a los editores de TEX, me temo que no están diseñados para ConTEXt. Aunque te suene raro, no te ayudarán. No sólo porque te costará mucho más adaptarlos a ConTEXt, sino porque tienen muchos más elementos que te distraerán de la escritura. La única excepción es *TEXworks*. Ése es el único editor de ConTEXt que te voy a explicar cómo configurar.

## F TEXWORKS

*TEXworks* es probablemente el mejor editor de texto para aprender ConTEXt. Está disponible para *Windows*, *MacOS X* y *Linux*[84]. Es especialmente fácil de manejar y es claro. Sobre todo, está diseñado teniendo en cuenta a ConTEXt. Es necesaria una pequeña adaptación, pero te enseño cómo se hace.

Como con cualquier otro programa con licencia libre, lo único que tienes que hacer es descargar la versión para tu sistema operativo e instalarlo. Después, ejecuta el programa. Verás esto:




## G CONFIGURACIÓN DE TEXWORKS

Antes de empezar a escribir, tienes que configurar *TEXworks* para que trabaje con *ConTEXt Suite*. Para eso, selecciona Editar y después pincha en la última opción del menú que se despliega, Preferencias.

### COLOREADO DE CÓDIGO

La pestaña que te encontrarás abierta será la de Editor. En esa pestaña, los cambios necesarios son:

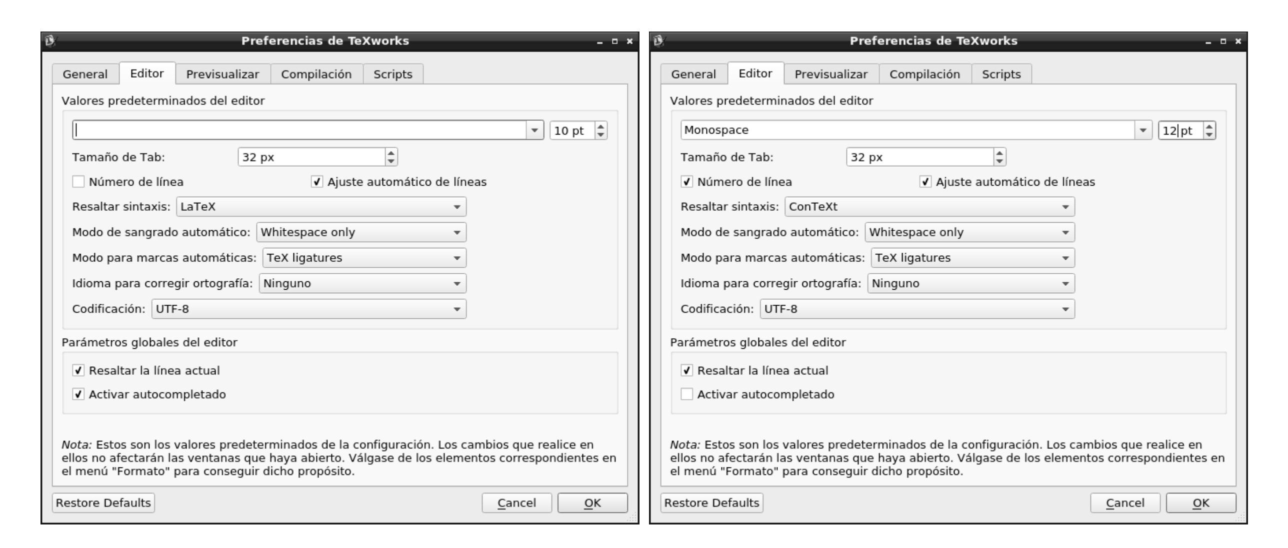
1. El tamaño del tipo de letra. Mejor ponerlo a doce puntos, porque el texto será más legible.

2. Activa el número de línea. Será mucho más fácil para saber dónde tienes alguna orden —o sobre todo, algún error—.

3. Selecciona ConTEXt en Resaltar sintaxis. Se coloreará mejor el código.

4. Elimina la última opción de Activar autocompletado. Evitarás errores tontos y volverte loco al principio.

La diferencia entre lo que te encuentras y lo que debes dejar es:




### COMPILACIÓN AUTOMÁTICA

Para que pulsando una combinación de letras ConTEXt genere el documento PDF del código que tienes en la ventana de *TEXworks*, es necesario que hagas lo siguiente. Sólo puede funcionar, si no te ha fallado *1.E. Comprobación final de funcionamiento*.

Dentro de la ventana de Preferencias, tienes que pasar a la ventana de Compilación. Ahí sólo tienes que hacer dos cosas. La primera es

mostrarle a a *TEXworks* dónde está ConTEXt. La segunda es indicarle cómo quieres que sea la orden de compilación.

La ventana de Compilación tiene dos partes. Tienes que empezar en la parte de arriba. Ahí indicarás a *TEXworks* la ubicación de ConTEXt. Para hacerlo, tienes que pulsar en el botón con el signo de suma. No puedes escribir, sino que tienes que ir abriendo diferentes carpetas.

Independientemente de tu sistema operativo, tienes que tener en cuenta:

1. No se trata de abrir ningún archivo, sino una carpeta.
2. De la ruta, vas abriendo carpetas. Cuando llegas a la carpeta final —de nombre `bin/`—, sólo tienes que presionar el botón Abrir.
3. Las carpetas están separadas por barras normales [85]. Lo que está antes de la barra contiene lo que está después de la barra.
4. Todas las versiones de *ConTEXt Suite* tienen una carpeta `bin/` en dentro de la carpeta principal. Aunque tenga el mismo nombre, no es esa carpeta. Depende de la versión del sistema operativo.
5. Aunque te mencione dos opciones por sistema operativo, sólo existirá una ruta.

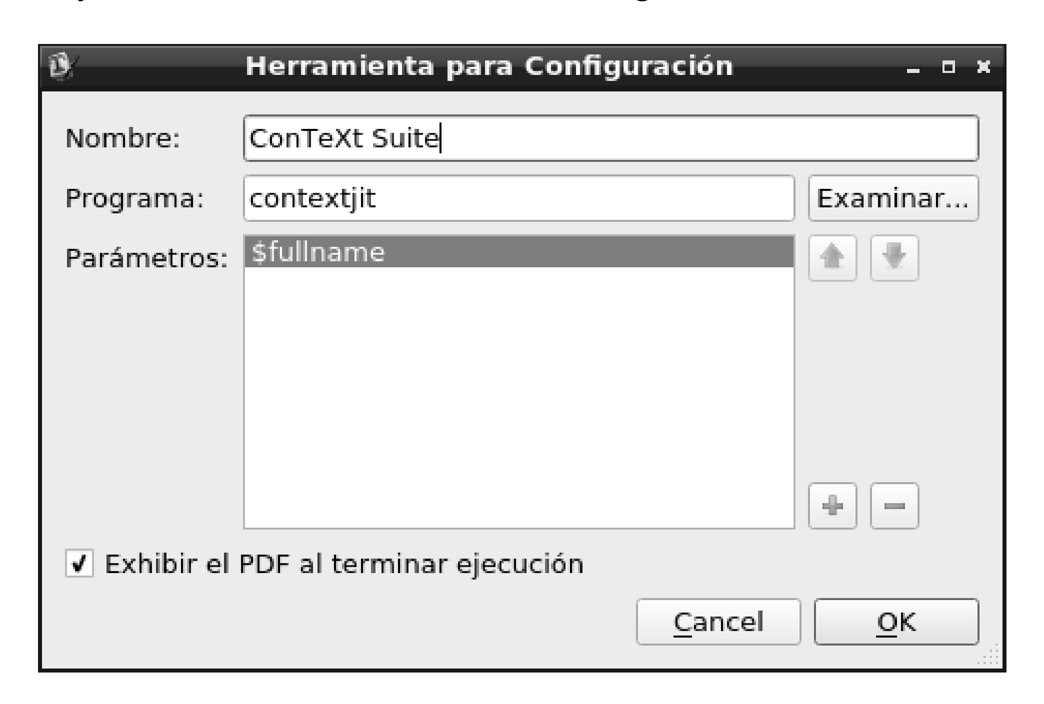
Si estás en *Windows*, ConTEXt lo hemos instalado en tu escritorio. En el diálogo que te da el programa, tienes que llegar a la carpeta de ConTEXt en el escritorio, para luego seleccionar `tex/texmf-mswin/bin/` o a `tex/texmf-win64/bin/` [86]. Cuando hayas llegado al final, tienes que pulsar Aceptar.

En *MacOS X*, has instalado ConTEXt en tu usuario. Tienes que ir a esa carpeta, para luego seleccionar `tex/texmf-osx-intel/bin/` o `tex/texmf-osx-64/bin/` [87]. Al llegar al final, aceptas.

Para *Linux*, *ConTEXt Suite* está instalado en tu usuario. Tienes que ir al directorio `context/tex/texmf-linux/bin/` o `context/tex/texmf-linux-64/bin/`. Cuando llegues, acepta.

Da igual el sistema operativo que uses, cuando hayas aceptado, tendrás una línea añadida en la ventana de Compilación. El texto que te mostrará será la ruta completa. Te aconsejo que la pinches una vez para seleccionarla y uses el botón de flecha arriba para colocarla como la primera opción de la lista [88].

Después debes crear una nueva herramienta. Tienes que pulsar en el botón de signo de suma de la parte baja de la ventana y añadir los datos tal y como te los muestro en la imagen:

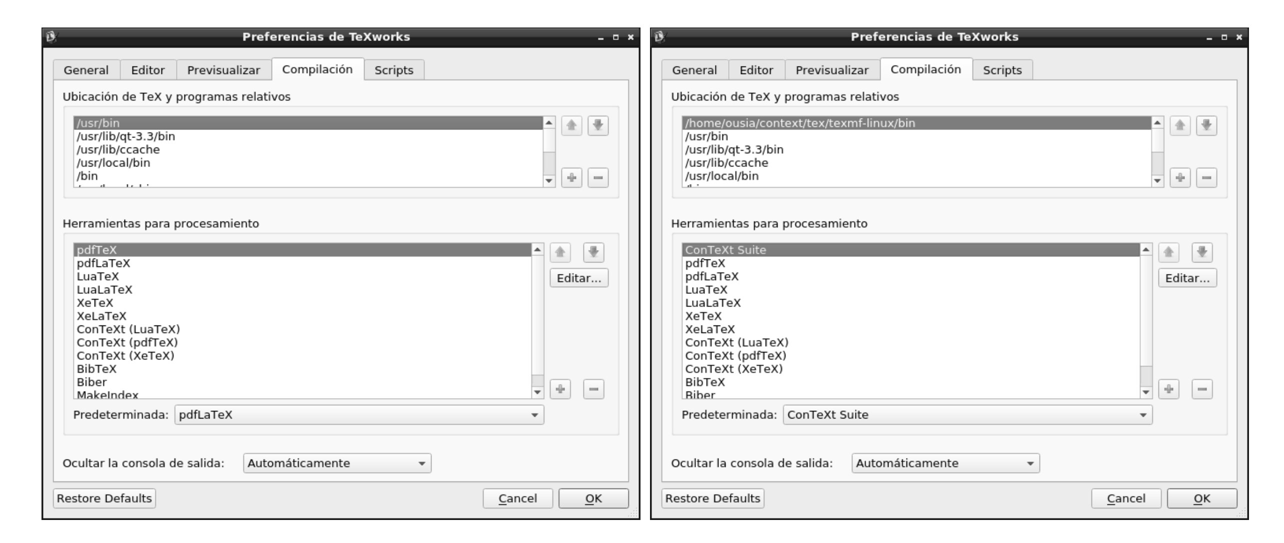



Todo tienes que teclearlo. Aunque para añadir parámetros, has de usar el botón de suma. Si te equivocas en un parámetro, puedes borrarlo con el signo de resta. Al pulsar aceptar [89], volverás a la ventana de Compilación.

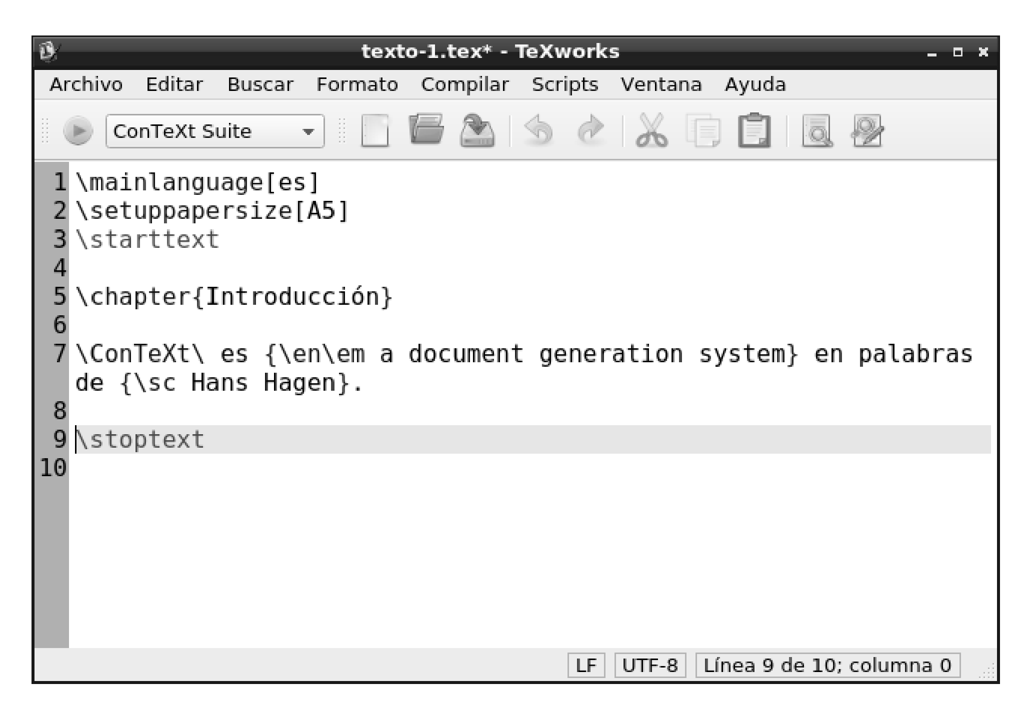
Puedes usar las flechas para subir *ConTEXt Suite* al primer lugar de la lista. Te será más cómodo si lo haces, pero no es totalmente

necesario. Sin embargo, es imprescindible que selecciones *ConTeXt Suite* como herramienta predeterminada del menú desplegable. Estará en el primer o en el último lugar de la lista que se te abrirá.

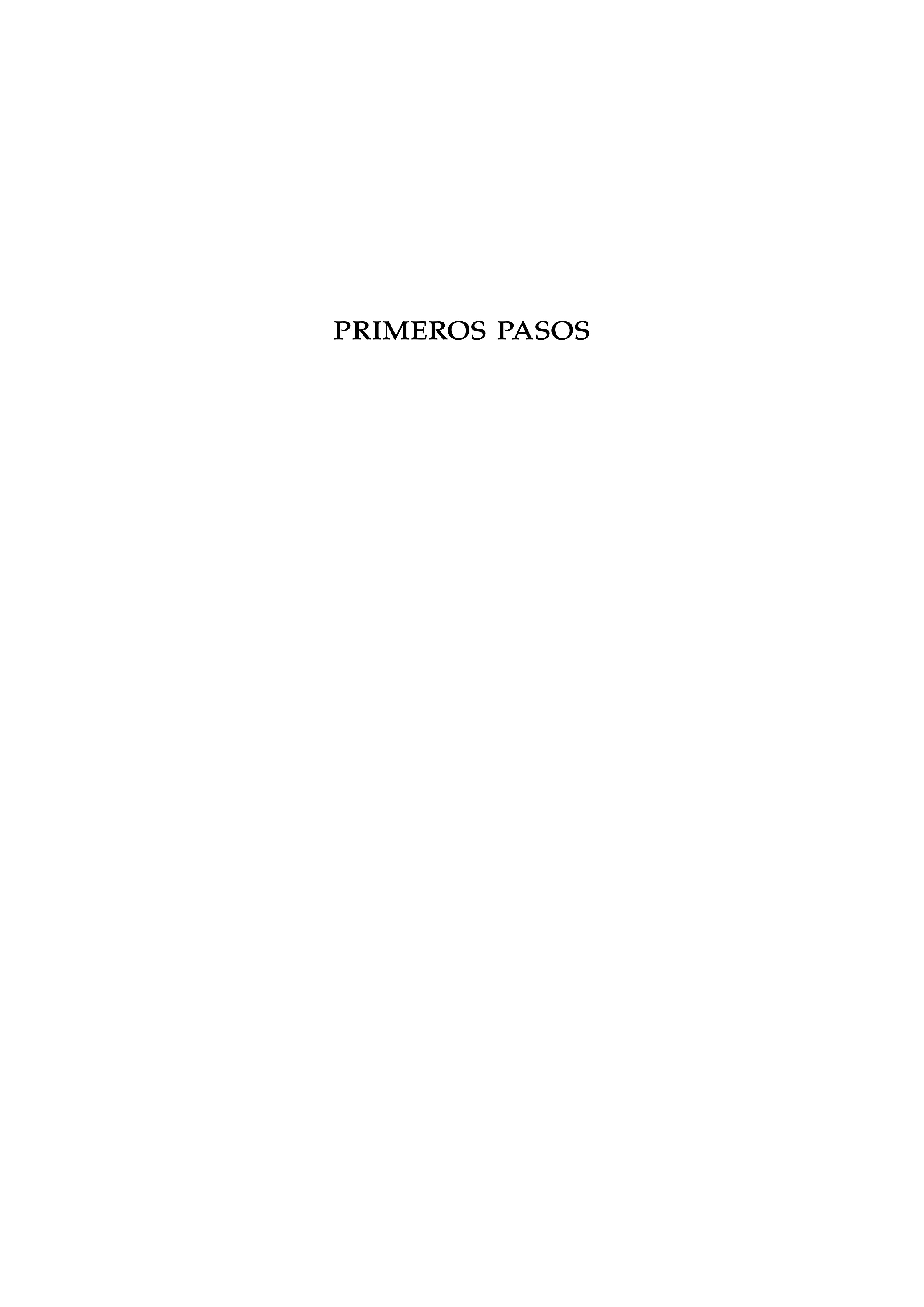
La diferencia entre lo que te encuentras y lo que debes tener sería:




Al aceptar los cambios, *TeXworks* ya estará listo para que lo puedas usar. La imagen final con texto sería:




Ya puedes empezar a dar tus primeros pasos con ConTeXt.



# PRIMEROS PASOS

# 5 EL PRIMER DOCUMENTO

*A Escribir algo sencillo B Nombres de documentos C Resultados D Detección de errores*

## A ESCRIBIR ALGO SENCILLO

Para escribir tu primer documento en ConTEXt, sólo tienes que iniciar *TEXworks*. Se abrirá la ventana de un documento nuevo. Si tuvieses otro editor de código, abre un documento nuevo. Una vez abierto, teclea:

```
\starttext
Estoy aprendiendo a usar \ConTeXt.
\stoptext
```

Para que el documento PDF se genere correctamente, tienes que copiar bien las órdenes. Si cambias el texto que no son órdenes, te saldrá otro texto que el de arriba. Pero no te dará errores. Si cambias las órdenes, puede que tengas fallos graves. Aunque no tengas miedo a cometer errores, es la manera de aprender. Igual que no aprendiste a andar sin caerte, es lógico que no te salga todo perfectamente bien en ConTEXt a la primera. Si fuese al revés, no necesitarías aprender.

Con *TEXworks*, lo único que tendrás que hacer después de teclear el pequeño ejemplo de arriba, es pulsar Ctrl+T[90]. Así pedirá que guardes el archivo, proponiéndote un nombre. Será `texto-1.tex`, aunque puedes cambiar el nombre y el lugar donde grabas el archivo. Una vez que esté guardado, comenzará la creación del documento PDF. Como verás luego, sólo te pedirá que confirmes la grabación del archivo cuando

no exista en el disco duro. Si existe, guardará los datos sin preguntar y empezará a compilar.

La compilación te aparecerá del siguiente modo. Aunque la primera vez pueda impresionar, no falla nada. En la parte baja de la ventana de texto se abrirá una ventana con mensajes. No tienes que entender nada. De hecho, los mensajes irán demasiado rápidos como para leerlos. Una vez que ConTEXt haya acabado la compilación, te mostará el documento PDF en una ventana al lado.

Si usar combinaciones de teclas no te gusta, puedes pulsar el botón verde fluorescente de la izquierda para comenzar la compilación. Si te fijas, se convertirá en rojo con un aspa mientras aparezcan los mensajes. Durante la compilación —la generación del documento PDF—, tienes que esperar a que ConTEXt haga su trabajo.

Los mensajes de la compilación desaparecen cuando ConTEXt ha podido crear el documento PDF sin problemas. Sólo si hay un error —como veremos en *5.D. Detección de errores*—, la ventana se quedará abierta. Por tanto, si aparece el documento PDF en la ventana de al lado, todo ha ido bien. Por cierto, el visor del documento PDF de *TEXworks* no es perfecto. Sirve para lo que está pensado, que veas en pantalla el resultado. Aunque para imprimir, es mejor que uses tu visor de documentos PDF habitual.

Éste ha sido tu primer documento en ConTEXt. Como no va a ser el único, es importante que tengas en cuenta unas consideraciones para seguir guardando más archivos. Porque los nombres y lugares que te da *TEXworks*, sólo te sirven para hacer la primera prueba. El archivo lo puedes borrar en cuanto termines de ver su resultado.

## B NOMBRES DE DOCUMENTOS

En este libro, te encontrarás muchos ejemplos de código que tendrás que teclear y compilar. Como ya sabes, es la manera de aprender.

Además, empezarás a usarlo para hacer tus cosas. Pueden ser pruebas o documentos finales. Aunque sean obviedades, te recomiendo que leas y consideres las siguientes recomendaciones para guardar archivos con código ConTEXt. Pueden ahorrarte algún problema y bastantes quebraderos de cabeza.

- El nombre del archivo debe ser significativo de su contenido. Sólo es relevante `archivo.tex` si es el único archivo que tienes de ConTEXt. Como con el resto de archivos, si los nombramos bien, nos ahorraremos tener que abrirlos para que saber qué contienen. En el peor de los casos, evitaremos pérdidas tontas por borrarlos pensando que tenían otro contenido.

- Debes acabar todos los archivos que contengan textos en ConTEXt con la extensión `.tex`. En principio, *TEXworks* siempre añadirá esa extensión cuando guardemos archivos de código. De ese modo, evitarás que la apertura con un programa distinto corrompa el archivo[91].

- Es mejor no usar espacios en los nombres de archivo, sino guiones o guiones bajos. De otro modo, la generación del documento PDF se puede complicar innecesariamente en ciertos casos.

- Es mejor no usar mayúsculas en los nombres, porque si bien *Unix* distingue entre mayúsculas y minúsculas, *Windows* no lo hace[92]. Para evitar problemas es mejor usar todo en minúscula, excepto en nombres propios.

- Lo mismo pasa con acentos en los nombres de archivo: mejor evítalos. Porque si cambias de sistema operativo es fácil que los pierdas o se corrompan[93].

- Sé ordenado con las carpetas de archivos. Los nombres de éstas también deben ser significativos. Además, cada carpeta debe contener carpetas que lógicamente le correspondan.

Como siempre, se trata de recomendaciones generales que considero que son buenas prácticas. Por supuesto, no son obligaciones. En algunos casos incluso puede ser recomendable no seguirlas. Pero mientras no entiendas qué ventaja obtienes con dejarlas de lado, es mejor que no olvides los inconvenientes que tendrás si te olvidas de ellas.

## C RESULTADOS

Cuando pulsas el botón verde fluorescente o la combinación Ctrl+T, se inicia la complicación. ConTEXt crea el documento PDF, procesando las órdenes del código del archivo. En la parte de órdenes, va mostrando una serie de mensajes [94]. Si se quedasen abiertos los mensajes [95], sólo te interesaría el último:

```
system          | total runtime: 12.838 seconds
```

Ése es el tiempo total que ha necesitado ConTEXt para generar el archivo PDF. Sobre ese tiempo, tienes que tener en cuenta lo siguiente:

- Mi portátil ya tiene una docena de años. Es muy difícil que tu ordenador no consiga un tiempo mucho menor.

- La primera vez que ConTEXt se ejecuta en un sistema, el propio programa necesita hacer la configuración interna [96]. Después, esa compilación necesita menos de cinco segundos en mi portátil

Como verás en el directorio en el que estés, ConTEXt crea tres archivos, además del que has guardado como `texto-1.tex`:

- `texto-1.pdf`
- `texto-1.tuc`
- `texto-1.log`

El primer archivo es el documento PDF final. De los otros dos, no vamos a hablar de momento. Son archivos que ConTEXt necesita para trabajar. El resultado es el documento PDF. Si lo abres, verás que el número de página está en el centro del encabezado. La tipografía es *Latin Modern Roman*. Es la predeterminada en ConTEXt, como *Times New Roman* en *Microsoft Word*, o *Liberation Serif* en *LibreOffice Writer*.

## D DETECCIÓN DE ERRORES

Existen tres certezas en la vida. En realidad, pueden tomarse como tres constantes. Son nada más y nada menos que la muerte, los impuestos y los fallos al escribir código informático. No pretendas ser infalible. Parte del aprendizaje incluye reconocer y solucionar los fallos, propios y ajenos. Por ejemplo, es muy común copiar `\startext` en vez de `\starttext`. Comerse una letra es común, sobre todo como en este caso, que la orden tiene una palabra que acaba y otra que empieza por la misma letra. Por si te sirve de consuelo, a mí me pasa a veces —más de lo que me gustaría—.

Para aprender a reconocer errores en el código —que ConTEXt no puede compilar—, tomemos el código del ejemplo siguiente. Abre *TEXworks*, copia el texto y compílalo. Ya sabes, tienes que pulsar el botón verde o Ctrl+T. Puedes grabarlo con el nombre de archivo `error.tex`.

```
\starttext
Estoy aprendiendo a usar \ConText.
\stoptext
```

El error consiste en escribir `\ConText`, en vez de `\ConTeXt`. Son órdenes distintas `\Context` o `\CONTEXT`, aunque la segunda pueda

tener el mismo resultado[97]. No debes olvidar nunca que mayúsculas y minúsculas forman órdenes distintas para un ordenador, aunque para las personas sean la misma palabra[98].

Si usamos mal las órdenes, el resultado será distinto del que esperamos. Si ConTEXt no es capaz de reconocer qué le mandamos, nos dará un mensaje de error. Cuando ConTEXt detecta un error en el archivo que está compilando, se para. Es un programa informático, no entiende lo que lee y se bloquea. El final del mensaje que se queda en la ventana es muy parecido a éste[99]:

```
tex error   > tex error on line 2 in file texto-1.tex:
    ! Undefined control sequence

l.2 Estoy aprendiendo a usar \ConText
                                    .

1     \starttext
2 >>  Estoy aprendiendo a usar \ConText.
3     \stoptext
```

Aunque te pueda parecer incomprensible, es muy sencillo. Es un mensaje claro.

1. En primer lugar, muestra que hay un error. Te dice dónde está, en qué línea se encuentra. Aunque te diga el nombre de archivo, sólo puede el que has compilado y estás escribiendo en *TEXworks*.

2. *Undefined control sequence* significa «orden desconocida».

   En ConTEXt, todas las órdenes comienzan con la barra invertida. Si no la conoce, tendrás siempre este error.

3. Te muestra el contenido de la línea y su contexto, numerados.

> Es importante tener las líneas numeradas, para saber dónde están los errores que puedas tener. Cualquier documento relevante de ConTEXt tendrá más de diez líneas [100].

Para corregir el error, tendrías que cambiar `\ConText` por `\ConTeXt` y volver compilar. Tanto si pulsas el botón como las teclas Ctrl+T, tienes que hacerlo dos veces. La primera es para anular la compilación anterior. Porque en realidad ConTEXt había detenido la compilación. La segunda vez es para compilar de nuevo la versión corregida.

# 6 ESCRITURA BÁSICA DE DOCUMENTOS



En este capítulo, empezamos a ver cómo se escribe un documento en ConTEXt. Son reglas básicas que se aplican a cualquier documento. Antes de seguir, es importante que te adviertas que desde esta página comienza tu aprendizaje en serio. Para que puedas avanzar lo mejor y más rápido posible, considera el contenido del apéndice *A. Aprender ConTEXt en la práctica*.

## A ESPACIOS

Los espacios en ConTEXt son únicos. Da lo mismo que escribas uno o varios seguidos: sólo se tomará como si hubiese uno. Las dos líneas del ejemplo siguiente tienen el mismo efecto.

```
\starttext
\ConTeXt\ es increíble.

\ConTeXt\     es            increíble.
\stoptext
```

Por supuesto, esto sólo se muestra en el resultado final, no en el código. De este modo, se facilita la escritura de documentos. Porque en un procesador de textos, hay en muchas ocasiones en que pueden escribirse dos o más espacios seguidos. Eso genera una separación desigual entre palabras. Especialmente cuando son sólo dos espacios seguidos, hace falta un ojo muy entrenado para detectar y poder corre-

gir ese fallo. La mayoría de personas lo reconoce con cierta dificultad cuando se lo han señalado.

Aunque sea más o menos común usando procesadores de textos, las separaciones entre textos no se hacen con varios espacios en ConTEXt. Tendremos que usar las órdenes adecuadas, pero no hay posibilidad de recolocar un fragmento de texto añadiendo espacios. No los necesitaremos.

### ESPACIOS DESPUÉS DE ÓRDENES

Las órdenes son la secuencia de letras que viene después de la barra invertida —como `\ConTeXt`—. Terminan una orden: cualquier signo de puntuación, una cifra, un signo que no sea una letra o un número —cuyo uso no esté reservado por el sistema—, o un espacio.

La particularidad es que ConTEXt ignora el espacio que viene después de la orden, por lo que si escribimos algo después del espacio, nos lo pondrá junto.

```
\starttext
\ConTeXt es increíble.

\ConTeXt\ es increíble.

\ConTeXt{} es increíble.
\stoptext
```

La manera de evitar que se obvie el espacio después de la orden es terminar la orden. El modo de hacerlo es crear un grupo vacío —un par de llaves `{}`—. O si se da un salto de línea justo al terminar la orden. O también se puede usar la orden `\␣` —barra invertida seguida de espacio—, que genera un espacio.

Es importante no olvidar nunca esta particularidad tras usar una orden, porque el texto quedará mal —faltará un espacio—.

# B PÁRRAFOS

Los párrafos en ConTEXt se forman por todas las líneas seguidas sin una una línea en blanco. Por tanto, un párrafo no se forma con un salto de línea, sino con al menos dos. Es lo que crea una línea en blanco. Al igual que con los espacios, con las líneas en blanco da igual que sean una o doscientas[101]. Al salto de línea al que le sigue texto tiene el mismo efecto que una línea en blanco. Si usas el signo de comentario —`%`— al final del último carácter de la línea, no tendrás ese espacio añadido.

Lo vemos en un ejemplo:

```
\starttext
Esto
es un
único párrafo.

Y aquí está el segundo.



Y el tercero.
\stoptext
```

En algunos casos, puede que necesites tener un salto de párrafo sin una línea en blanco en medio—. Para eso, existe la orden `\par`. Como muestra el ejemplo:

```
\starttext
Primer párrafo.\par Segundo párrafo.

Tercer párrafo.\par Cuarto párrafo.
\stoptext
```

Al igual que las líneas en blanco, da lo mismo que uses `\par` una o varias veces. O que lo mezcles con líneas en blanco. Tendrás un nuevo párrafo, pero no más separación entre ellos.

### SALTOS DE PÁRRAFO Y DE LÍNEA

Un salto de párrafo se marca por una línea en blanco entre una línea con texto y la siguiente que contenga texto o con la orden `\par`. El contenido del párrafo es todo lo que esté entre dos líneas en blanco, o que no contenga la orden la orden `\par`. El párrafo se reconoce por la sangría o el espacio vertical que pueda tener.

Puede haber ocasiones en que quieras tener una línea nueva dentro de un mismo párrafo. Esa línea no tendrá sangría o espacio vertical separándola del resto del párrafo. Por defecto, ConTEXt no sangra los párrafos ni añade espacio vertical de separación entre ellos. La línea la introduces con la orden `\\`, que es equivalente a `\crlf`.

```
\setupwhitespace[big]
\starttext
Éste es un párrafo\\
Ésta es una línea.

Ése es otro párrafo
\stoptext
```

### MARCAR TODOS LOS SALTOS DE LÍNEA

Puede haber un algún caso en que nos interese que se marquen todos los saltos de línea. El ejemplo más claro es poesía, pero podría haber otros. Por supuesto, podemos poner `\crlf` o `\\` al final de cada línea. Pero para este caso existe `\startlines…\stoplines`. Así podremos copiar y pegar directamente de otros archivos.

En el ejemplo 102 se muestra como ambas posibilidades tienen casi el mismo efecto:

```
\starttext
Érase un hombre a una nariz pegado,\\
érase una nariz superlativa,\\
érase una alquitara medio viva,\\
érase un pez espada mal barbado;

era un reloj de sol mal encarado,\\
érase un elefante boca arriba,\\
érase una nariz sayón y escriba,\\
un Ovidio Nasón mal narigado.

\startlines
Érase el espolón de una galera,
érase una pirámide de Egipto,
las doce tribus de narices era;

érase un naricísimo infinito,
frisón archinariz, caratulera,
sabañón garrafal, morado y frito.
\stoplines
\stoptext
```

La diferencia está en que el documento no tiene definido espacio entre párrafos. Es la opción predeterminada de ConTEXt. Lo puedes arreglar añadiendo `\setupwhitespace[line]` como primera línea en el archivo de código. Más adelante, verás cómo especificar el espacio entre párrafos.

## C COMENTARIOS

Puede haber partes del archivo de ConTEXt que no quieras que aparezcan en el documento final. Tanto texto, como órdenes para el

documento. Por el motivo que sea, puede interesarte poder leerlas en el archivo de código. Podrán ser anotaciones sobre el texto mismo, incluso sobre el código, o tareas pendientes de escritura.

Los comentarios son la manera de incluir texto que ConTEXt ignorará cuando genere el documento PDF. El carácter que introduce el comentario es el signo de porcentaje —%—. Es válido desde donde se introduzca hasta el final de la línea. Corta la línea desde de su aparición, sin introducir un espacio después.

```
\starttext
Eres lo más b%asto.
ello.

% Si escribes lo primero,
% no ganarás muchas amistades.

% Ni mucho menos, amores.
\stoptext
```

Si tienes que convertir en comentario una línea de texto completa, no hay más problema en introducir el signo de porcentaje al comienzo de la línea. *TEXworks* permite hacerlo automáticamente pulsando las teclas Ctrl+Mayús+] [103]. Sólo lo hace automáticamente desde el comienzo de línea. Si tienes varias líneas seleccionadas, todas serán comentarios. El modo de hacer que vuelvan a ser texto normal —para que dejen de ser comentarios— es la combinación de teclas Ctrl+Mayús+\ [104]. Usando los bloques de comentarios, puedes obviar una parte importante de un documento, pero sin tener que borrarla —por si la necesitásemos después—.

De los comentarios se sigue una consecuencia importante. Si necesitamos que el final de una línea no añada un espacio antes de la palabra de la siguiente línea, sólo tenemos que poner un signo de porcentaje al final de la línea. Como en el ejemplo siguiente:

```
\starttext
Esta es una línea
y esta es otra, aunque forman un
párrafo.

Esta es otra lín%
ea que no añade espacio al final.
\stoptext
```

## D PREÁMBULO Y TEXTO

Todos los documentos de ConTEXt tienen que tener el bloque propio del texto. Se define por la pareja de órdenes `\starttext` y `\stoptext`. Lo hemos visto en ya unos cuantos ejemplos. Esa estructura de comienzo y final del texto es lo que marca la estructura básica del documento.

Con el primer ejemplo de este libro:

```
\mainlanguage[es]
\setuppapersize[A5]
\starttext

\chapter{Introducción}

\ConTeXt\ es {\en\em a document generation system} en
palabras de {\sc Hans Hagen}.
\stoptext
```

El texto comienza con `\starttext`. Antes de esa orden, sólo está la configuración del documento. Podemos llamarlo el preámbulo del documento. En esa parte se especifican las condiciones generales que se aplicarán al documento en su conjunto. Incluso aunque puedas escribirlas dentro del bloque de texto es muy importante que toda la configuración esté en el preámbulo —antes de `\starttext`—.

En el preámbulo, sólo puede haber órdenes. Antes de `\starttext`, no debe haber texto, únicamente puede haber órdenes de configura-

ción. El texto que pongas antes de `\starttext`, ConTEXt lo ignorará. Si quieres escribir texto en el preámbulo, te aconsejo que lo pongas como comentario. Aunque por pura legibilidad de tu documento, es mejor que lo hagas después de `\stoptext`.

Cada orden puede formar un parrafo. Esto es, puede contener saltos de línea, pero no tener una línea en blanco en medio. No es bueno que mezcles órdenes distintas en la misma línea. No fallará ConTEXt —si están bien escritas—, pero es fácil que te equivoques tú. En el preámbulo se aplica también lo dicho en las secciones *6.A. Espacios* y *6.B. Párrafos*. Entre diferentes órdenes, puede haber tantos saltos de línea como queramos[105].

Tras `\stoptext` se acaba el documento. Por supuesto, podemos escribir lo que queramos ahí. Estará en el archivo origen, pero no aparecerá en el documento que genere ConTEXt.

## E ESCRIBIR CÓDIGO

Como verás a lo largo de todo este libro, el código de ConTEXt puede escribirse de muchas maneras. Todas pueden ser correctas. La corrección consiste en que no haya errores en la lectura que tiene que hacer ConTEXt. Es quien tiene que compilar el código que se encuentre en el archivo. Lo único que necesita el programa es no encontrar errores en las órdenes. Así podrá terminar la generación de documento PDF sin problemas.

Sin embargo, un archivo de código lo tiene que poder leer y modificar una persona. La gran mayoría de las veces serás tú quien modificarás un archivo tras haberlo teclear por primera vez. Para facilitarte la legibilidad y la rápida comprensión futuras del texto y el código —en el archivo están entrelazados—, es bueno que tengas en cuenta lo siguiente:

- Debes escribir siempre las órdenes con las opciones separadas por un espacio después de la coma.

- Las órdenes de configuración, que están en el preámbulo, es importante agruparlas —con una línea en blanco antes y después— según aspectos. Por ejemplo, los grupos de órdenes podrían ser: idiomas, diseño de página, tipografías, títulos, sumarios, índices, documento interactivo, órdenes propias…

- Es bueno escribir una línea por párrafo. No es obligatorio, pero creo que es más fácil trabajar así [106].

- Separar los párrafos de texto con una única línea entre ellos.

- Los apartados de un documento —partes, capítulos, secciones…— puede ser bueno que las separes con dos o tres líneas antes del título de cada apartado. Cuanto más texto contenga el apartado, más líneas debe tener. Pero siempre el mismo número líneas de separación para el mismo tipo de apartado.

Estas recomendaciones —o similares— hacen más fácil tu trabajo con documentos ConTEXt. O no lo entorpecen innecesariamente. Si no sigues estas recomendaciones, no tendrás errores. No son obligaciones. Pero tendrás más dificultad a la hora de detectar y corregir incluso tus propios errores. Porque te será más difícil leer el código que está mezclado con el texto.

## F COMPARTIR UN ARCHIVO DE CONTEXT

Una idea básica y recurrente al comienzo del aprendizaje es que los archivos de ConTEXt estén escritos de la manera más clara posible. Básicamente, para hacerte lo más fácil posible su lectura. Tanto por el texto, como principalmente por las órdenes que incluyas. Porque

con el paso del tiempo, puedes encontrarte tratando de descrifrar un jeroglífico en lo que habías pensado que estaba total y absolutamente claro en el momento de teclearlo.

Esta consideración también es importante si tienes que compartir archivos de ConTEXt. La claridad es definitiva cuando quieras que alguien te ayude con tus archivos. Porque si no se entiende el código —o su lectura se le hace muy difícil porque es poco claro—, es fácil que le hagas desistir de la noble intención a casi toda persona que quiera echarte una mano.

Por último, es importante una consideración final. Con los comentarios, hemos visto la posibilidad de incluir texto en el código que no va a reflejarse en el documento final. Eso tiene una consecuencia importante. Si vamos a compartir documentos, es mejor no hacer anotaciones en los archivos de todo lo que no sea el documento mismo. Para eso es mejor siempre usar otros archivos. Porque puedes acabar teniendo una indiscrección que lamentes.

# APÉNDICES

# A APRENDER CONTEXT EN LA PRÁCTICA

Como has podido ver hasta ahora, ConTEXt es un conjunto de órdenes que se mezclan con el texto para poder obtener como resultado documentos PDF. Si se trata de órdenes —dicho muy llanamente—, has de conocer cómo puedes usarlas en los archivos de código. No es cuestión de que aprendas una lista de órdenes de memoria, sino de que sepas cómo funciona cada orden. De otro modo, no te servirán para nada en absoluto.

Soy consciente de que parte de lo que te voy a contar no serán ideas nuevas. Te volveré a explicar cosas ya vistas, porque es importante que las entiendas. Sólo quiero que la dedicación a este libro no acabe volviéndose una pérdida de tiempo. Porque no aprendas nada y quieras dejarlo, o porque no aprendas tan rápido como podrías.

Desde que dejas el primer ejemplo —en el comienzo del capítulo *6. Escritura básica de documentos*—, todos los ejemplos de código son prácticas. Esto es, son textos para que teclees en *TEXworks* —u otro editor, si es tu caso—, compiles el código y veas el resultado del documento PDF. Es la única manera de aprender con este libro.

Por si no estuviese claro, es esencial que lo hagas, porque eso es usar ConTEXt. Cuando sepas, lo harás con los textos y con las composiciones tipográficas que quieras. Al principio, tendrás que teclear y ver que pasa. Es importante que pruebes. Ver qué pasa cuando cambias algo. En el peor de los casos, no te saldrá lo que esperabas.

Es importante que experimentes, que trastees, para ver qué consigues con cada cosa. Tienes que ver por tí mismo qué funciona en tus pruebas y qué no. Con más práctica, sabrás qué cosas pueden llegar a funcionar y cuáles no. ¿Cómo? Después de haber probado mucho.

Aunque te parezca arcaico —incluso antipedagógico—, no copies y pegues el código de este libro. Escribe, que en un ordenador se traduce como «teclea». Si copias y pegas tendrás menos posibilidades de aprender, de memorizar y de entender. No todo es memoria, pero de nada sirve si no recuerdas las órdenes. Tienes mucha mayor exposición a lo aprendido tecleando que copiando y pegando.

Si no fuese absolutamente necesario que teclease, yo mismo te lo diría. Te voy a poner un ejemplo tonto de esa necesidad. Como esta orden, hay otras muchas y no tan sencillas. Como veremos dentro de muy poco, el texto de un documento tiene que marcar dónde empieza y dónde acaba. Para eso, existen las órdenes `\starttext` y `\stoptext`. Si las copias y las pegas, el día que las tengas que escribir, las teclearás mal. De vez en cuando, en la lista de usuarios de ConTEXt hay un comentario sobre `\starttext` escrita como `\startext` o parecido. Son personas con un conocimiento avanzado, pero no son máquinas. El ejemplo es de la orden más básica. Imagínate cómo puede ser el resto de errores y no te olvides de teclear todo el código.

Sólo queda un último asunto, la atención que es decisiva para aprender. En el aprendizaje, se mezclan memoria y comprensión. No entenderíamos nada si no pudiésemos recordar. Pero tampoco la memoria es mero almacenamiento de datos. Cada vez es más común que las personas nos comparemos a máquinas. Personalmente creo que es una batalla perdida y un error. Aunque a lo que aquí me refiero es a lo que podría llamarse «modo multitarea». De modo similar a los ordenadores, las personas haríamos varias cosas al mismo tiempo.

Si estás con una fuente de sonido y/o imagen externa, atendiendo al correo electrónico, con los mensajes del teléfono móvil e intentando aprender ConTEXt, es imposible que te dificultes mucho más tu proceso de aprendizaje. La dificultad no surge sólo cuando acometes todo simultáneamente. Tampoco te ayudaría nada la interrupción periódica

de la edicación a ConTEXt cada poco tiempo por uno de esos estímulos externos.

Las máquinas pueden desarrollar multitarea porque su rendimiento es plano. Tardarán más o menos tiempo, pero el rendimiento final será igual. Las personas tenemos más o menos atención y nuestro desempeño varía. Aunque parezca lo contrario, nuestro desempeño en casi cualquier actividad es proporcional a la atención que le prestamos al realizarla. Psicológicamente, la multitarea nos parece altamente gratificante, si bien nuestro desempeño cae en picado y sin que siquiera lo advirtamos [107].

Nuestro grado de atención para realizar una actividad lo mejor posible depende de qué hagamos. Con un ejemplo básico, ir en bicicleta no necesita quizá toda nuestra atención. Pero si no tenemos la suficiente, acabaremos en un accidente. La lectura exige toda nuestra atención. Leer no es pasar páginas sin saber qué hemos leído. Este libro es más exigente que la lectura de una novela. Exige toda nuestra atención. ¿Qué pasa si le prestamos menos? Como poco, que aprenderemos menos y más lentamente.

Mi consejo personal es que el tiempo que puedas dedicar a leer y practicar los contenidos de este libro, lo hagas con dedicación exclusiva. No tiene que ser mucho tiempo, sólo tiene que ser sin distracciones. Porque tu provecho será mucho mayor.

Como resmuen, no dudo de que se trata de tu aprendizaje personal. No sólo tú pones las reglas, sino que tienes que ponerlas. Lo que tienes delante es un libro, no una persona. Aunque las recomendaciones contenidas en este apéndice pueden desempeñar alguna función en el resultado final.

# B DESCOMPRESIÓN DE ARCHIVOS



## 1 LA IMAGEN DE LA CAJA

La compresión de archivos es un modo muy habitual de manejar archivos. Creo que la imagen más adecuada[108] para entender qué es comprimir un archivo puede ser el ejemplo de meter objetos en una caja. O mejor aún, meter ropa en una maleta o en una caja.

Comprimimos los archivos para que ocupen menos espacio. Del mismo modo, guardamos la ropa en cajas para que nos robe el menor espacio de nuestras casas. Los archivos informáticos se comprimen para que sea más fácil transportarlos. Al igual que metemos la ropa en maletas para que sea más fácil llevarla. En el caso de los archivos, la mayor facilidad no está en la facilidad de transporte, sino en su menor tamaño. Como los archivos se transmiten y se copian a una velocidad constante, cuanto más pequeño sea el archivo, antes lo tendremos en el destino que queramos.

Al igual que la ropa en caja o en maletas, no podemos usar directamente archivos comprimidos. Ni siquiera podemos saber qué hay dentro de un archivo comprimido —al igual que dentro de una maleta o caja, que suelen ser opacas— sin abrirlo antes. Del mismo modo que con alguna caja o maleta, existen archivos comprimidos que necesitan una clave para que los podamos abrir. Pero los archivos comprimidos que necesitamos para lo que trata este libro no tienen contraseña.

Existen diferentes métodos de comprimir archivos. Al igual que existen diferentes modos de hacer maletas o de llenar cajas con ropa.

Para ver qué hay dentro de un archivo comprimido, tenemos que saber cuál es el formato de compresión que usa. Al igual que para ver los contenidos de una maleta o una caja tenemos que saber tienen organizados sus contenidos. En el caso de archivos comprimidos, son programas informáticos los que ven lo que hay dentro. Pero para que los programas puedan manejar un archivo comprimido tienen que poder leer el formato de compresión en el que está comprimido el archivo.

Descomprimir es extraer el archivo o archivos que están dentro del archivo comprimido. Igual sacar la ropa de una maleta y colocarla en un armario. Para eso necesitamos un programa informático. Y al igual que nos muestra los contenidos de un archivo comprimido, extrae los archivos que tiene dentro.

## 2 PROGRAMAS PARA DESCOMPRIMIR

No estoy totalmente seguro, pero creo que todos los sistemas operativos incorporan un programa para descomprimir archivos en los formatos más comunes. Hasta donde sé, al menos todos son capaces de descomprimir formato .

En *Windows*, existe un buen compresor llamado *7-Zip*[109]. Descomprime muchos más formatos que los que comprime. Y una vez instalado, nos permite descomprimir archivos con el menú que aparece al pulsar sobre un icono con el botón derecho del ratón. Existen otros compresores, pero este es de código abierto y gratuito.

En *MacOS X*, existe un descompresor —no comprime— capaz de manejar una gran cantidad de formatos de compresión, *The Unarchiver*[110]. Es un descompresor casi universal.

De ese proyecto, existe una utilidad de descompresión con órdenes escritas, para ambos sistemas operativos, llamada `unar`[111]. Aunque es más fácil usar una utilidad gráfica para descomprimir

# C OPCIONES DE IMPRESIÓN DEL DOCUMENTO PDF



Probablemente una de las propiedades más útiles de los documentos PDF es poder incluir opciones de impresión en el propio documento. Creo que esas opciones son decisivas cuando vas a una copistería con una impresión que no sea una página DIN A4 o carta en papel del mismo tamaño. En mi experiencia, cuando te sales de eso y te lo tienen que hacer, es fácil que algo falle. Personalmente, me he olvidado alguna vez de decir algo, me han entendido mal o lo han hecho mal. No es cuestión de echar la culpa a nadie. Nadie gana nada con impresiones mal hechas, incluso aunque no te las cobren.

Para esos casos —o cuando hay que mandar imprimir documentos enviados por correo electrónico— existen unas opciones de impresión que pueden añadirse al documento PDF. Estas características están definidas en la especificación PDF [112].

ConTEXt incorpora cuatro de las opciones de impresión posibles en el documento PDF [113]:

- Impresión a doble cara.
- Tamaño de papel.
- Reducción o ampliación de impresión.
- Número de copias.

Como veremos en las siguientes secciones, son opciones de la orden `\setupinteractionscreen`. Las tres primeras son parte de la misma opción y puede mezclarse entre sí. La cuarta es una opción propia.

## 1 IMPRESIÓN A DOBLE CARA

La impresión a doble cara tiene tres posibilidades. La primera es la impresión por una única hoja. Dependiendo del documento, puede ser importante que sólo se imprima por una cara. Así se desactiva la impresión a doble cara cuando la impresora la tiene preseleccionada. La segunda posibilidad es doble cara con el giro en vertical. Es el modo en que se imprimen a doble cara la mayoría de documentos. Por último, queda la impresión a doble cara con giro horizontal. Sirve la para la impresión por ambas caras en apaisado.

La opción de `\setupinteractionscreen` es `option`, con los siguientes valores posibles:

`page` imprime a una sola cara, ignorando la impresión a doble cara si está predeterminada en la impresora.

`portrait` doble cara con giro vertical.

`landscape` doble cara con giro horizontal.

## 2 TAMAÑO DE PAPEL

*Adobe Acrobat* imprime el tamaño de papel que sea en el que tenga por defecto. En Europa, el papel es DIN A4. Entiendo que en América, será en tamaño carta. Esto es independiente del tamaño de papel que tenga el documento. Para conseguir imprimir en un tamaño de papel distinto —por ejemplo, DIN A5—, es necesario especificar que tome el tamaño del documento PDF.

La orden es `\setupinteractionscreen[option=paper]`. Como he dicho antes, esta opción es compatible con la impresión a doble cara o la reducción o ampliación de impresión.

## 3 REDUCCIÓN DE IMPRESIÓN

*Adobe Acrobat* tiene varias opciones respecto a la reducción o ampliación de la impresión. Las más comunes son las referidas a los márgenes de impresión. ¿Qué son los márgenes de impresión? La distancia desde el borde del papel en que impresora no puede imprimir. Suele ser un centímetro. Si no se respeta el margen de impresión, no saldrá en la hoja lo que esté dentro de él.

Una manera de evitar ese fallo es reducir el contenido de la página para que quepa dentro de los márgenes de impresión. Si seleccionamos esta opción, el contenido se reducirá, aunque no sea necesario porque no haya nada situado en el margen de impresión. De este modo, el contenido de la página sale más ligeramente más pequeño. Con papel europeo, en DIN A4 apenas se nota, pero en tamaños menores la diferencia es más apreciable.

El modo de evitar esa reducción innecesaria es desactivar la reducción de la página. El modo de hacerlo es con `\setupinteractionscreen[option=fixed]`. De ese modo, la reducción a los márgenes de impresión será nula en el documento en que incluyamos la orden.

## 4 SELECCIÓN DE VARIAS OPCIONES

Como ya he dicho antes, es posible seleccionar varias opciones de impresión. Sin embargo, tanto la impresión a doble cara o sencilla, como tomar el tamaño de papel del documento PDF, añaden la opción de desactivar la reducción del contenido de la página a los márgenes de impresión.

Los siguientes grupos de opciones son equivalentes entre sí:

```
\setupinteractionscreen
    [option=paper]
\setupinteractionscreen
```

```
    [option={page, paper}]
\setupinteractionscreen
    [option={fixed, page, paper}]

\setupinteractionscreen
    [option=page]
\setupinteractionscreen
    [option={fixed, page}]

\setupinteractionscreen
    [option=portrait]
\setupinteractionscreen
    [option={fixed, portrait}]

\setupinteractionscreen
    [option=landscape]
\setupinteractionscreen
    [option={fixed, landscape}]
```

La manera de hacer que las propiedades de impresión sean independientes es añadir como primer valor. De este modo, deshabilitamos en todas las opciones [114]. El ejemplo sería:

```
\setupinteractionscreen
    [option={none, page}]

\setupinteractionscreen
    [option={none, portrait}]

\setupinteractionscreen
    [option={none, landscape}]
```

De este modo, tendríamos impresión a una o doble cara —con giro vertical u horizontal—. La reducción a los márgenes de impresión no está especificada en el documento PDF. En la práctica, dependerá de las opciones de impresión de *Acrobat*.

## 5 NÚMERO DE COPIAS

Es posible que de algún documento necesitemos varias copias del mismo. *Adobe* permite que puedan especificarse en el mismo archivo PDF hasta un máximo de cinco copias. Se logra mediante la orden `\setupinteractionscreen[copies=5]`.

# NOTAS

1 Para un uso más avanzado, es posible usar archivos XML como archivos de origen. Pero el uso más avanzado supone el uso más básico, puesto que las órdenes de composición tipográfica son de ConTEXt. En una sola frase, para usar ConTEXt con XML, hay que saber usar antes ConTEXt.

2 Su explicación se encuentra en *Tipografía digital*.

3 No puedo dejar de citar el origen de la expresión, que es el libro *LATEX, una imprenta en sus manos* de Bernardo Cascales.

4 Lo advierto porque no pasa nada si no se entiende. La explicación contiene información relevante, pero se irá tratando progresivamente.

5 Por si alguien tiene mucho interés, lleva por título *Seminumerical Algorithms*.

6 "I had spent 15 years writing those books, but if they were going to look awful, I didn't want to write any more." D. E. Knuth, *Digital Typography*, Center for the Study of Language and Information, p. 5.

7 A pesar de que *Acrobat Reader* tiene un mecanismo por el que interpreta muchos textos como enlaces. Independientemente de que pueda fallar en algún caso, debemos incluir los hiperenlaces en el documento, para que puedan usarse con otros visores de documentos PDF.

8 Como, por ejemplo, `git`.

9 Obvio el ejemplo más común de la composición de fórmulas matemáticas, porque es un campo desconocido para mí.

10 Además, podemos, de modo totalmente automático, ordenar a ConTEXt que añada toda la información sobre marcas de integridad de los archivos adjuntos. Igual que puede lograrse de modo automático de una clase, pueden generarse de varias clases.

Explico por qué puede resultar útil lo anterior. Si no lo entiendes, puedes ignorarlo totalmente. No sólo se trata de que la marca de integridad de archivos SHA1 no sea ya segura —como muestra `https://shattered.io`—. Sin apenas

esfuerzo añadido, porque todo lo hace el programa, se pueden añadir también marcas SHA224, SHA256, SHA384 y SHA512. Si la seguridad de alguna queda comprometida, las marcas de mayor nivel garantizan la integridad de la marca de menor nivel.

No hace falta ser analista de seguridad informática para darse cuenta de que lo que hoy es seguro, mañana es fácil que pueda no serlo. Porque se investiga más y porque cada vez es más fácil tener más capacidad de proceso informático. Uno de los mayores expertos mundiales en seguridad informática —Bruce Schneier, `https://www.schneier.com`— afirma que la seguridad es un proceso. Otra manera de decirlo, es afirmar que la seguridad informática es una escalera y tenemos que estar siempre en el siguiente peldaño.

11 En igual situación se encuentran aquellas personas en la posición contraria. No se trata de descalificaciones mutuas, sino simplemente de poder dialogar sobre capacidades de ambos programas.

12 Las condiciones de la licencia pueden verse en `https://www.gnu.org/licenses/gpl-2.0.html`.

13 La base de esta elección es también de legibilidad. El inglés carece de acentos o de otras marcas diacríticas —como la diéresis o la marca de nuestra *ñ*—. Las comillas inglesas ocupan la parte alta de las vocales, como en “come in!”. En español, «camión» es más legible. De este modo, las comillas no invaden el espacio natural de acentos y otras marcas diacríticas.

También sucede en otras lenguas. Como el francés, con tres acentos. En alemán, las comillas son inversas. También tiene marcas diacríticas —aunque esos dos puntos marquen letras diferentes—, como en »schön«. Usar las más comunes —como en „Frühstück“— no mejora la legibilidad.

14 La razón del uso de la cursiva es el aviso a quien lee para que se prepare para algo distinto. El propio cambio de alfabeto es un signo inconfundible de cambio. ¿O acaso no se percibe que χαλεπὰ τὰ καλά es una expresión en idioma distinto?

En cambio, los títulos de obras deben estar en cursiva siempre.

15 Al igual que en el caso anterior, que las citas han de estar siempre entrecomilladas, independientemente del alfabeto en que estén escritas.

16 La cosa se puede complicar más diciendo que la ß sólo se usa en Alemania y Austria, no en Suiza o Liechtenstein —allí sería *Strasse*—.

En todo caso, la *ß* —una ligadura de dos *s*, la forma de la primera es *ſ*— no tiene nada que ver con la *B* latina o la $\beta$ griega.

17 "[Y]ou are designing not the black marks on the page, but the space in between"; E. SPIEKERMANN, *One Thing Only*, en `http://spiekermann.com/en/one-thing-only/`.

18 Acrónimo de la denominación *Portable Document Format*.

19 Pares de letras que en las tipografías *Garamond* o *Times New Roman* están ligados, mientras *Palatino* no lo están. Depende del diseño del tipo de letra, si bien la mayoría de tipografías con serifas tienen esas ligaduras.

20 Este fallo en concreto lo he encontrado en libros de otras editoriales también prestigiosas.

21 Uso deliberadamente ejemplos de lo que no puede hacer un procesador de textos.

22 Un comentario de Ramiro Santa Ana me hace notar que la expresión es un españolismo. El *Diccionario de la lengua española* explica la expresión —`http://dle.rae.es/?id=A28RNZ3#Ez2ICjj`— como tener que solucionar algo por sí. En este caso, la expresión pide a ConTEXt que componga el texto por sí mismo. Precisamente en eso consiste un tipógrafo digital.

23 Por si alguien duda, el texto de la nota va entre las llaves.

24 Puede verse una versión en directo en `https://www.youtube.com/watch?v=8Pa9x9fZBtY`. No estoy interesado en gustos musicales. Simplemente no cabe afirmar que Mark Knopfler no sabe tocar la guitarra eléctrica.

25 Hay que teclear para aprender a escribir las órdenes. Copiar y pegar es más cómodo, pero a la larga nos pasará factura.

26 Siempre cabe descargar el código de ConTEXt y empezar a leerlo para ver cómo funciona. Sería entender ConTEXt desde dentro. Aunque si puedes hacerlo, no pierdas el tiempo leyendo este libro.

27 Antes de incluir un fragmento completo de código, lo he ejecutado y he visto que funciona como se espera. Si no fuese así, te pediría que abrieses una incidencia como explico en *6.C. Comentarios*.

28 Abordaré la edición digital en *Aprender* `pandoc`*. Una introducción a la edición digital*, en `http://www.aprender-pandoc.tk`.

29 Una mínima muestra de su finalidad está descrita en *C. Opciones de impresión del documento PDF*. El desarrollo de capacidades de visualización es secundario respecto a la impresión.

30 Para quien tenga más inclinación técnica: es irreal numerar títulos con contadores. Porque podría llegar a funcionar teóricamente sólo si todo el texto está en un único archivo `.xhtml` dentro del documento ePub. Cualquier dispositivo gasta demasiados recursos con sólo abrir un documento ePub así.

31 Como *iBooks* de *Apple*, que la añaden a todos los elementos.

32 En realidad, uso un único archivo de origen para generar también el documento PDF. Aunque en este caso, `pandoc` genera automáticamente un archivo que ConTEXt compila en un documento PDF. El modo de hacerlo lo explicaré en *Aprender* `pandoc`.

33 Un ejemplo básico, si lees cualquier idioma que use el alfabeto cirílico —ruso, búlgaro o ucraniano, entre otros—, el sistema no te mostrará información sobre los libros en ese alfabeto. Y cada libro debe incluir tipos de letra para que se vea el texto cirílico. Otro ejemplo sería el griego politónico, que la ortografía griega anterior a 1982. El griego antiguo usa también la ortografía politónica. También lo usa todo el griego moderno y casi todo el griego contemporáneo.

34 La explicación técnica es que *Adobe Reader Mobile* no reconoce pseudoclases. Por tanto, un selector en CSS del tipo `:lang(de)` no funciona.

35 De nuevo la explicación técnica es que *Adobe Reader Mobile* no acepta pseudoelementos, como `:before` o `:after`, ni el atributo `content` en las hojas de estilo en cascada.

36 En *[Microtipografía]*.

37 Por supuesto, no es cuestión de añadirlos al escribir el texto. Lo tiene que hacer el programa que genere el documento PDF. Dos dificultades añadidas

son las siguientes. Si el texto contiene pasajes en otros idiomas, es importante que los guiones ocultos sean correctos según el idioma de las palabras en las que se encuentran. Y puede haber elementos del texto —títulos o párrafos centrados, por ejemplo— en los que no queramos que las palabras se partan al final de la línea.

38 Como es obvio, las críticas al libro son libres. Las explicaciones son más efectivas que las descalificaciones. No conviene olvidar el respeto debido a las personas.

39 "If I have seen further, it is by standing on the shoulders of giants." I. Newton, *Letter to Robert Hooke*.

40 La imagen de la viñeta que se encuentra en `http://wiki.contextgarden.net/Humour#Super_Wolfgang` describe su capacidad de respuesta.

41 En realidad, no empecé a usar ConTEXt o TEX, sino LATEX.

42 `http://live.contextgarden.net/`.

43 `https://www.tug.org/texlive/`, `https://www.tug.org/mactex/`, `https://www.tug.org/protext/` o `http://www.miktex.org/`.

44 Hans Hagen hace el trabajo duro con ConTEXt y también lidera un grupo reducido de personas que hacen posible que ConTEXt esté disponible para toda persona que quiera usarlo.

45 Es una distribución de TEX con versiones nuevas cada año.

46 De broma se suele decir que hay tres certezas en la vida: la muerte, los impuestos y los fallos al escribir código informático. Suelen ser menores y suelen corregirse con una rapidez media que me gustaría que fuese el estándar en muchos servicios por los que se paga manteniento informático.

47 Por supuesto, están hechas en el mismo ordenador.

48 Es sólo un ejemplo de hace bastante tiempo. El modo en que trabajo el libro, hace mucho más engorrosa la comparación.

49 Alguna de las versiones las guardo para comprobar cambios entre versiones de ConTEXt. En realidad, al tratarse de *1.C. Una distribución portátil*, no hay problema en tener cientos de versiones. La única limitación es el espacio

de almacenamiento. Cuando lo necesite, borraré la mayoría de versiones de fechas más antiguas.

50 De modo resumido, así funcionan la mayoría de programas.

51 Otro programa de ejecución tipográfica sería pdfTEX. Es en el programa en que LuaTEX se basa parcialmente. Aunque para ConTEXt, está ya obsoleto.

52 El portátil que he usado tiene más de una década, si bien usa una versión actualizada de *Linux*. La conexión a internet es de fibra óptica, con descargas a un máximo real de seis megabytes por segundo.

53 Por supuesto, dentro del mismo sistema operativo. Los programas de un sistema operativo no pueden ejecutarse directamente en otro sistema operativo distinto.

54 Excepto los archivos que hayamos creado, como es obvio.

55 Si no fuese así, consulta el apéndice *B. Descompresión de archivos*.

56 La instalación en todo el sistema y no sólo para un usuario, está explicada en `http://wiki.contextgarden.net/ConTeXt_Standalone#System-wide_installation`.

57 La lista completa de los programas está en `http://wiki.contextgarden.net/ConTeXt_Standalone#Dependencies`. Desaconsejo instalar todos los programas. Porque dedicaremos tiempo a algo que probablemente no necesitaremos.

58 `http://minimals.contextgarden.net/setup/context-setup-mswin.zip`. Si el procesador tiene arquitectura de 64bit, el archivo sería `http://minimals.contextgarden.net/setup/context-setup-win64.zip`. Sólo si estamos seguros, debemos usar la segunda opción.

59 `%USERPROFILE%` es una variable que da el directorio del usuario que está usando el sistema. `Desktop\` es el escritorio y `context\` donde debería estar descomprimido el archivo `context-setup-mswin.zip` —o en su caso, `context-setup-win64.zip`—.

60 Es algo que dejado de hacer por problemas de espacio. Sencillamente copiaba la carpeta `context` a `context-fecha` —como en `context-16102014`—. Y actualizaba la *ConTEXt Suite* del directorio `context`.

61 `rsync` viene instalado en *MacOS X*. Agradezco a José Ramón Portillo de la lista ES–TEX la explicación. Si hubiese algún error, es única y exclusivamente mío.

62 En *MacOS X*, en realidad sería:



```
mkdir ~/context/ && cd ~/context/ && rsync -ptv
rsync://contextgarden.net/minimals/setup/first-setup.sh . &&
./first-setup.sh --modules=all && source tex/setuptex &&
mtxrunjit --generate && cd
```

El añadido de las tipografías del sistema en *MacOS X* viene de fábrica. No es necesario activarlo.

63 Por eso no aparece esa orden en la versión para *MacOS X*.

64 Te recuerdo que ConTEXt necesita tener LuaTEX instalado para funcionar.

65 Si no te da miedo la ventana de órdenes, yo uso la siguiente:



```
curl -I http://pragma-ade.com/context/beta/cont-tmf.zip
```

Sólo pide los datos de archivo de la última version de pruebas. En el momento que escribo, sería:

```
HTTP/1.1 200 OK
Date: Tue, 27 Jun 2017 19:45:28 GMT
Server: Apache
Last-Modified: Tue, 27 Jun 2017 16:13:32 GMT
ETag: "61a0042-459565f-552f3595198d8"
Accept-Ranges: bytes
Content-Length: 72963679
Content-Type: application/zip
```

Así sé cuándo puedo actualizar la versión de desarrollo. Para evitar impaciencias, es diferente *date* —momento en que se hace la consulta— de *last modified* —cuándo se publicó la última versión de desarrollo—.

66 Mejor hacerlo así que con una orden escrita. Porque si hay algún error, las consecuencias pueden ser fatales.

67 La lista completa de sistemas es:

```
freebsd freebsd-amd64 linux linux-64 linux-armhf mswin
openbsd openbsd-amd64 osx-64 osx-intel osx-ppc solaris-intel
solaris-sparc win64
```

68 Básicamente, los ordenadores son grandísimas calculadoras binarias.

69 También llamada «comando». Considero es un barbarismo, porque *command* como sustantivo es «orden» en español.

70 Como puede resultar ambiguo, lo explico. `\ConTeXt.` es la orden que escribe el logotipo ConTEXt seguida de un punto. El punto no es parte de la orden, sino que le sigue —como cualquier otro signo de puntuación—.

71 La literatura especializada la denomina «argumento». Independientemente de la excesiva cercanía al ingles *argument*, me parece que es mejor evitar esa denominación. Porque con un cierto espíritu crítico, la presunta terminología se revela como un empobrecimiento innecesario de nuestra lengua.

72 "Computers are good at following instructions, but not at reading your mind"; D. KNUTH, *The TEXbook*, Addison–Wesley, p. 9.

73 Por si alguien tiene curiosidad, le invito a hacer una copia de un documento `.odt` o `.docx`, a cambiarle la extensión a `.zip`, a descomprimir los contenidos del archivo y a ver qué encuentra.

74 Para poder comprobarlo, no hay más que ir a cualquier artículo de *Wikipedia* y seleccionar la pestaña Editar código.

75 Disponible en `http://pandoc.org`.

76 No cabe usar esta opción en vez de aprender ConTEXt. Por la sencilla razón de que con pandoc podremos generar archivos ConTEXt, pero seremos incapaces de cambiar su formato.

77 Los colores variarán entre los diferentes editores de código. Lo que no debe variar es qué está coloreado y qué no.

En el documento PDF, no muestro colores, sino tonos de gris. La razón es no distraer excesivamente de la lectura del texto.

78 Por supuesto, dependiendo del editor puede ser una tecla única —como F9— o una combinación de teclas —como Ctrl+T—. Dependiendo del editor, esa asignación de teclas incluso se puede cambiar.

79 Como norma general, es bueno que vayas grabando los cambios cada poco tiempo, conforme vayas redactando el texto.

80 `http://geany.org/`. En principio está disponible para *Windows* y *Linux*. Puede llegar a usarse en *MacOS X*, `http://wiki.geany.org/howtos/osx/running`.

81 Disponible en `https://notepad-plus-plus.org/`.

82 Por si alguien tiene curiosidad, son órdenes de compilación más complejas que las que trataré aquí. Están descritas en la sección *8.G A Single Keystroke* de *From* `pandoc` *to ConTEXt: A Method to Generate High–Quality PDF Documents*, disponible en `http://www.from-pandoc-to-context.tk`.

83 En realidad, el editor que voy a explicar —*TEXworks*— se ha diseñado teniendo *TEXShop* como modelo. Si quieres usarlo, lo único que tienes que buscar es cómo configurarlo. Aunque creo que no son precisamente para principiantes, tienes instrucciones en `http://wiki.contextgarden.net/Mac_Installation#Setting-Up_TeXShop_for_ConTeXt_Single_User_Installation`. Lo siento, pero sólo puedo explicar un programa que funcione también en mi sistema.

84 En `https://www.tug.org/texworks/`. Tiene la misma licencia que ConTEXt.

85 Incluso en *Windows*, que usa la barra invertida. En ConTEXt, la barra está reservada exclusivamente para las órdenes. Para separar directorios y subdirectorios, incluso en *Windows*, se usa la barra normal.

86 Necesitarás la primera ruta si lo que usaste para descargar ConTEXt lo conseguiste de la dirección `http://minimals.contextgarden.net/setup/context-setup-mswin.zip`. Será la segunda ruta, si la utilidad que has usado es `http://minimals.contextgarden.net/setup/context-setup-win64.zip`.

87 Si tienes todavía un ordenador con architectura PPC, la ruta sería `tex/texmf-osx-ppc/bin/`. No puedo asegurar que *TEXworks* funcione con ese procesador.

88 Si tienes instalado *TEX Live*, es la única manera de que *ConTEXt Suite* se use en primer lugar.

89 Por si te lo preguntas, en mi imagen es posible que el botón sea OK, porque mi sistema está en inglés. Para explicarlo, he tenido que cambiar la configuración de idioma en *TEXworks*.

90 Da lo mismo que el teclado estés en mayúsculas o minúsculas para pulsar la combinación de teclas. Para escribir texto o código, siempre tienes que distinguir entre mayúsculas y minúsculas.

91 Por supuesto, si quieres distinguir los archivos de ConTEXt de los de TEX o LATEX podrías usar la extensión `.mkiv`. Pero es importante que sepas qué haces y por qué. Lo único que tendrías es que asignar la extensión `.mkiv` a *TEXworks*.

92 Es bueno evitar —como recojo en este consejo y el siguiente— cuestiones básicas que puedan dar problemas en otro sistema operativo. Que tú no uses otro sistema operativo no quiere decir nada. Puedes tener que usarlo temporalmente, o puedes cambiar en el futuro. O tener que trabajar con otras personas que usen otros sistemas distintos del tuyo. Además, las minúsculas son más legibles que las mayúsculas. Especialmente en una pantalla.

93 El nombre de archivo puede verse afectado si contiene caracteres distintos de letras —no acentuadas— y números. En principio, nada impide que nombres un archivo `introducción.tex` o incluso `περὶ-ἑρμηνείας.tex`, al menos en *MacOS X* o *Linux*. Sin embargo, cuando copies esos archivos a *Windows*, esos nombres de archivo pueden darte problemas. Está claro que si lo evitas, no tendrás dificultades añadidas.

94 En mi portátil, los mensajes pueden verse de otro modo en `https://asciinema.org/a/5xxQpP2GF3fX0q48X3F5GeXJe`. Aunque es *Linux*, en otro sistema operativo sería similar.

95 En realidad, si quieres que se muestre la parte de los mensajes —que se llama consola—, pulsa Ctrl+\. Para volverla a cerrar, vuelve a pulsar la misma combinación de teclas.

96 En realidad, tiene que hacer un listado de cada uno de los archivos de ConTEXt con sus rutas en el dispositivo correspondiente. También necesita

más tiempo la primera vez que tiene que buscar los tipos de letra en el sistema operativo. Lo veremos en su momento.

97 En este caso concreto, existe la orden `\CONTEXT` que se comporta igual que `\ConTeXt`. De no ser así, `\CONTEXT` daría error. Que haya dos maneras de escribir una orden es una excepción. No sucede con el resto de órdenes. Sólo existe un único modo de escribir la mayoría de órdenes en ConTEXt.

98 Como ya he explicado en *2.C. Mayúsculas y minúsculas en el código*.

99 Está ligeramente editado para la línea quede bien en el documento PDF. No he borrado ninguna parte importante.

100 Además, también puedes ir a la línea escribendo su número, si pulsas Ctrl+L en *TEXworks*. Ir a un determinado número de línea es una función común en los editores de textos.

101 La única regla es la legibilidad de nuestros propios archivos ConTEXt.

102 Francisco de Quevedo, *A un nariz*, en *Parnaso español* 249, disponible en `https://es.wikipedia.org/wiki/A_una_nariz`.

103 Siento que la combinación de teclas sea tan complicada. De hecho, en *Linux* no funciona, porque Mayús+] da la marca diacrítica ˚. Sirve para la combinación en letras como *å* e *ů*, con sus respectivas mayúsculas. En *Geany*, el editor que uso habitualmente, la combinación de teclas para comentarios es Ctrl+E.

104 Al menos en *Linux*, la combinación de teclas Mayús+] da la marca ¯. Sirve para teclear las letras como *ā*, *ē*, *ī*, *ō* y *ū*, con sus mayúsculas correspondientes. En *Geany*, la combinación de teclas para quitar el comentario es la misma que para ponerlo, Ctrl+E.

105 En este caso, ni siquiera tiene que haber una línea en blanco. La única regla es la legibilidad de tu código. Es especialmente importante, porque las órdenes de configuración del preámbulo son más difíciles de leer que el texto mezclado con órdenes. La única razón es que no son texto común.

106 Si partes las líneas dentro de cada párrafo, siempre que modifiques el párrafo, te quedará mal partidas las líneas. A no ser que uses un editor de programación.

107 Una explicación de este fenómeno y otros relacionados se encuentra en S. Turkle, *Reclaiming Conversation: The Power of Talk in a Digital Age*, con información disponible en `http://reclaimingconversationbook.com`. Traducida como *En defensa de la conversación. El poder de la conversación en la era digital*, con información en `http://aticodeloslibros.com/index.php?id_product=106&controller=product`.

108 No pretendo que mi explicación sea técnicamente exhaustiva y totalmente verdadera. Hasta donde yo sé, el ejemplo no cojea. No obstante, lo más importante de este ejemplo y de esta explicación es que haga comprensible qué es un archivo comprimido y cómo podemos usarlo.

109 `http://7-zip.org/`.

110 `http://unarchiver.c3.cx/unarchiver`.

111 `http://unarchiver.c3.cx/commandline`.

112 Adobe Systems Inc., *Document management — Portable document format — Part 1: PDF 1.7*, pp. 364–365, en `https://wwwimages2.adobe.com/content/dam/Adobe/en/devnet/pdf/pdfs/PDF32000_2008.pdf#page=364`. Desde un punto de vista técnico, es curioso que estén desarrolladas como preferencias del visor —*entries in a viewer preferences dictionary*—.

113 Existen más opciones de impresión, siendo quizá la más destacable el intervalo de páginas —`/PrintPageRange` en el código PDF—. Es posible que Hans Hagen acepte desarrollar esa propiedad, si alguien se lo pide explicándole las razones de su utilidad. Aunque es sencillo rehacer un documento PDF con las páginas necesarias para la impresión.

114 Es la única manera de mantener la compatibilidad con los valores anteriores.

*Aprender ConTEXt. Una introducción a la tipografía digital* está compuesto tipográficamente con ConTEXt (`http://contextgarden.net`) por mediación de `pandoc` (`http://pandoc.org`).

El método empleado está recogido en *From `pandoc` to ConTEXt: A Method to Generate High–Quality PDF Documents* (`http://www.from-pandoc-to-context.tk`).

Los fundamentos teóricos de este enfoque se describen en *Un solo origen. Un método automatizado de edición electrónica* (`http://www.un-solo-origen.tk`).